macromedia®

FLASH®

Referencia del lenguaje de componentes

8

Agradecimientos

Dirección del proyecto: Sheila McGinn

Redacción: Bob Berry, Jen deHaan, Peter deHaan, David Jacowitz, Wade Pickett

Dirección de la edición: Rosana Francescato

Redactora jefe: Lisa Stanziano

Edición: Evelyn Eldridge, Mary Ferguson, Mary Kraemer, Jessie Wood

Dirección de la producción: Patrice O'Neill, Kristin Conradi, Yuko Yagi

Producción y diseño multimedia: Adam Barnett, Aaron Begley, Paul Benkman. John Francis, Geeta Karmarkar, Masayo Noda, Paul Rangel, Arena Reed, Mario Reynoso

Reconocimiento especial a Jody Bleyle, Mary Burger, Lisa Friendly, Stephanie Gowin, Bonnie Loo, Nivesh Rajbhandari, Mary Ann Walsh, Erick Vera, Jorge G. Villanueva, los probadores beta y todo el equipo de ingeniería de Flash y Flash Player y los equipos de control de calidad.

Primera edición: septiembre de 2005

Macromedia, Inc.
601 Townsend St.
San Francisco, CA 94103, EE. UU.

Contenido

CAPÍTULO 1

Diccionario de componentes

1

En el manual *Referencia del lenguaje de componentes*, se describe cada uno de los componentes y su API (interfaz de programación de aplicaciones). Para aprender a utilizar, personalizar y crear componentes, consulte *Utilización de componentes*. En *Referencia del lenguaje de componentes*, la descripción de cada componente contiene la siguiente información:

- Interacción con el teclado
- Previsualización dinámica
- Accesibilidad
- Configuración de los parámetros del componente
- Utilización del componente en una aplicación
- Personalización del componente con estilos y aspectos
- Métodos, propiedades y eventos de ActionScript

Los componentes se presentan en orden alfabético, pero en las tablas siguientes están clasificados por categorías.

Este capítulo contiene las siguientes secciones:

Tipos de componentes

En las tablas siguientes se muestran los distintos componentes de la versión 2 de la arquitectura de componentes de Macromedia, clasificados por categorías.

Componentes de interfaz de usuario (IU)

Componente	Descripción
Componente Accordion (sólo en Flash Professional)	Conjunto de vistas verticales superpuestas con botones en la parte superior que permiten a los usuarios cambiar de vista.
Componente Alert (sólo en Flash Professional)	Ventana que presenta un mensaje y botones para capturar la respuesta de los usuarios.
Componente Button	Botón que puede cambiarse de tamaño y personalizarse con un icono personalizado.
Componente CheckBox	Permite hacer una elección booleana (true o false).
Componente ComboBox	Permite seleccionar una opción en una lista de desplazamiento con distintas opciones. Este componente puede tener un campo de texto seleccionable en la parte superior de la lista que permita a los usuarios buscar en la lista.
Componente DataGrid (sólo en Flash Professional)	Permite a los usuarios mostrar y manipular varias columnas de datos.
Componente DateChooser (sólo en Flash Professional)	Permite a los usuarios seleccionar una o más fechas de un calendario.
Componente DateField (sólo en Flash Professional)	Campo de texto no seleccionable con un icono de calendario. Cuando un usuario hace clic en el recuadro de delimitación de un componente, Macromedia Flash muestra un componente DateChooser.
Componente Label	Campo de texto no editable de una línea.
Componente List	Permite seleccionar una o varias opciones en una lista de desplazamiento.
Componente Loader	Contenedor de un cargador de archivos SWF o JPEG.
Componente Menu (sólo en Flash Professional)	Menú de aplicación de escritorio estándar; permite a los usuarios seleccionar un comando de una lista.
Componente MenuBar (sólo en Flash Professional)	Barra horizontal de menús.
Componente NumericStepper	Cuadro de texto con flechas en las que puede hacer clic para incrementar o disminuir un valor numérico.

Componente	Descripción
Componente ProgressBar	Muestra el progreso de un proceso, como una operación de carga.
Componente RadioButton	Permite seleccionar entre opciones que se excluyen entre sí.
Componente ScrollPane	Muestra clips de película, mapas de bits y archivos SWF en un área limitada con barras de desplazamiento automático.
Componente TextArea	Campo de texto de varias líneas opcionalmente editable.
Componente TextInput	Campo de texto de una línea opcionalmente editable.
Componente Tree (sólo en Flash Professional)	Permite manipular información jerárquica.
Componente Window	Ventana arrastrable con una barra de título, texto, borde, un botón Cerrar y área de presentación de contenido.
Componente UIScrollBar	Permite añadir una barra de desplazamiento a un campo de texto.

Gestión de datos

Componente	Descripción
Clases de vinculación de datos (sólo en Flash Professional)	Clases que implementan la funcionalidad de vinculación de datos en tiempo de ejecución de Flash.
Componente DataHolder (sólo en Flash Professional)	Almacena información y puede utilizarse como conector entre componentes.
Interfaz API de DataProvider	Modelo para listas de acceso lineal de datos; ofrece capacidades básicas de manipulación de matrices que difunden cambios de datos.
Componente DataSet (sólo en Flash Professional)	Bloque para crear aplicaciones gestionadas por datos.
Componente RDBMSResolver (sólo en Flash Professional)	Permite guardar datos en cualquier origen de datos admitido. Este componente convierte los datos XML que se pueden recibir y analizar mediante un servicio Web, JavaBean, servlet o página ASP.
Clases de servicios Web (sólo en Flash Professional)	Clases que permiten el acceso a los servicios Web que usan SOAP (Simple Object Access Protocol). Estas clases están en el paquete mx.services.

Componente	Descripción
Componente WebServiceConnector (sólo en Flash Professional)	Proporciona acceso sin script a llamadas de método de servicio Web.
Componente XMLConnector (sólo en Flash Professional)	Lee y escribe documentos XML mediante los métodos `GET` y `POST` de HTTP.
Componente XUpdateResolver (sólo en Flash Professional)	Permite guardar datos en cualquier origen de datos admitido. Este componente convierte el paquete delta en XUpdate.

Componentes multimedia

Componente	Descripción
Componente FLVPlayback (sólo en Flash Professional)	Permite incluir fácilmente un reproductor de vídeo en la aplicación Flash para reproducir vídeo transmitido de forma progresiva a través de HTTP desde Flash Video Streaming Service (FVSS) o desde Flash Communication Server (FCS).
Componente MediaController (sólo en Flash Professional)	Controla la reproducción de flujo de medios en una aplicación (véase "Componentes multimedia (sólo en Flash Professional)" en la página 861).
Componente MediaDisplay (sólo en Flash Professional)	Muestra los flujos de medios en una aplicación (véase "Componentes multimedia (sólo en Flash Professional)" en la página 861).
Componente MediaPlayback (sólo en Flash Professional)	Combinación de componentes MediaDisplay y MediaController (véase "Componentes multimedia (sólo en Flash Professional)" en la página 861).

Administradores

Clase	Descripción
Clase DepthManager	Administra la profundidad de apilamiento de los objetos.
Clase FocusManager	Gestiona el desplazamiento entre componentes mediante la tecla Tabulador. También maneja los cambios de selección cada vez que el usuario hace clic en la aplicación.
Clase PopUpManager	Permite crear y eliminar ventanas emergentes.
Clase StyleManager	Permite registrar estilos y administra los estilos heredados.

Clase	Descripción
Clase SystemManager	Permite gestionar la activación de una ventana de nivel superior.
Clase TransitionManager	Permite administrar efectos de animación en diapositivas y clips de película.

Pantallas

Clase	Descripción
Clase Form (sólo en Flash Professional)	Permite manipular pantallas de aplicación de formulario en tiempo de ejecución.
Clase Screen (sólo en Flash Professional)	Clase base para las clases Slide y Form.
Clase Slide (sólo en Flash Professional)	Permite manipular pantallas de presentación de diapositivas en tiempo de ejecución.

Otras listas de este capítulo

En este libro también se describen varias clases y API que no están incluidas en las categorías de componentes enumeradas en la sección anterior. Estas clases y API se enumeran en la siguiente tabla.

Elemento	Descripción
Interfaz API CellRenderer	Conjunto de propiedades y métodos que los componentes basados en listas (List, DataGrid, Tree, Menu y ComboBox) usan para manipular y mostrar contenido de celda personalizado para cada una de sus filas.
Interfaz Collection (sólo en Flash Professional)	Permite administrar un grupo de elementos relacionados, denominados elementos de colección. Cada elemento de colección de este conjunto tiene propiedades que se describen en los metadatos de la definición de clase del elemento de colección.
Clase DataGridColumn (sólo en Flash Professional)	Permite crear objetos para usarlos como columnas de una cuadrícula de datos.
Clase Delegate	Permite ejecutar una función pasada desde un objeto a otro en el contexto del primer objeto.
Interfaz Delta (sólo en Flash Professional)	Proporciona acceso a los cambios de objeto de transferencia, colección y nivel de objeto de transferencia.

Elemento	Descripción
Clase DeltaItem (sólo en Flash Professional)	Proporciona información sobre una operación individual realizada en un objeto de transferencia.
Interfaz DeltaPacket (sólo en Flash Professional)	Junto con la interfaz Delta y la clase DeltaItem, permite gestionar los cambios realizados en los datos.
Clase EventDispatcher	Permite añadir y eliminar detectores de eventos para que el código pueda reaccionar correctamente a los eventos.
Interfaz Iterator (sólo en Flash Professional)	Permite recorrer los objetos de una colección.
Clase MenuDataProvider	Permite que instancias de XML asignadas a una propiedad `Menu.dataProvider` usen métodos y propiedades para manipular sus propios datos así como las vistas de menú asociadas.
Clase RectBorder	Describe los estilos usados para controlar los bordes de componentes.
Clase SimpleButton	Permite controlar algunos aspectos de la apariencia y el comportamiento de un botón.
Interfaz TransferObject	Define un conjunto de métodos que los elementos administrados por el componente DataSet deben implementar.
Interfaz TreeDataProvider (sólo en Flash Professional)	Conjunto de propiedades y métodos usados para crear XML para la propiedad `Tree.dataProvider`.
Clase Tween	Permite utilizar el código ActionScript para mover, cambiar el tamaño y desvanecer clips de película fácilmente en el escenario.
Clase UIComponent	Proporciona métodos, propiedades y eventos que permiten a los componentes compartir comportamientos comunes.
Clase UIEventDispatcher	Permite a los componentes emitir determinados eventos. Esta clase se añade a la clase UIComponent.
Clase UIObject	Clase base de todos los componentes de la versión 2.

CAPÍTULO 2

Componente Accordion (sólo en Flash Professional)

2

El componente Accordion es un navegador que contiene una secuencia de elementos secundarios que se muestran de uno en uno. Los elementos secundarios deben ser objetos que hereden de la clase UIObject (que incluye todos los componentes y pantallas incorporados con la versión 2 de la arquitectura de componentes de Macromedia); generalmente, los elementos secundarios son una subclase de la clase View. Esto incluye clips de película asignados con la clase mx.core.View. Para mantener el orden de tabulación en los elementos secundarios de Accordion, éstos deben ser también instancias de la clase View.

Un componente Accordion crea y administra botones de encabezado en los que un usuario puede hacer clic para desplazarse por los elementos secundarios del componente. Tiene un diseño vertical con botones de encabezado que abarcan la anchura del componente. Hay un encabezado asociado con cada elemento secundario y cada encabezado pertenece al componente Accordion, no al elemento secundario correspondiente. Cuando un usuario hace clic en un encabezado, se muestra el elemento secundario asociado debajo de dicho encabezado. La transición al nuevo elemento secundario utiliza una animación de transición.

Un componente Accordion con elementos secundarios acepta una selección y cambia el aspecto de sus encabezados para mostrarla. Cuando el usuario usa el tabulador para desplazarse hasta un componente Accordion, el encabezado seleccionado muestra el indicador de selección. Un componente Accordion que no tenga elementos secundarios no aceptará selecciones. Al hacer clic en componentes seleccionables dentro del elemento secundario seleccionado, dichos componentes quedarán seleccionados. Cuando la instancia de Accordion esté seleccionada, podrá controlarla con las teclas siguientes:

Tecla	Descripción
Flecha abajo, flecha derecha	Desplaza la selección al siguiente encabezado de elemento secundario. La selección recorre los encabezados de elementos secundarios desde el último al primero sin cambiar el elemento secundario seleccionado.
Flecha arriba, flecha izquierda	Desplaza la selección al encabezado del elemento secundario anterior. La selección recorre los encabezados de los elementos secundarios desde el primero al último sin cambiar el elemento secundario seleccionado.
Fin	Selecciona el último elemento secundario.
Intro/Espacio	Selecciona el elemento secundario asociado con el encabezado que está seleccionado.
Inicio	Selecciona el primer elemento secundario.
Av Pág	Selecciona el siguiente elemento secundario. La selección recorre los elementos secundarios desde el último al primero.
Re Pág	Selecciona el elemento secundario anterior. La selección recorre los elementos secundarios desde el primero al último.
Mayús+Tabulador	Desplaza la selección al componente anterior. Este componente puede estar dentro del elemento secundario seleccionado o fuera del componente Accordion; nunca será otro encabezado en el mismo Accordion.
Tabulador	Desplaza la selección al componente siguiente. Este componente puede estar dentro del elemento secundario seleccionado o fuera del componente Accordion; nunca será otro encabezado en el mismo Accordion.

Los lectores de pantalla no pueden acceder al componente Accordion.

Utilización del componente Accordion (sólo en Flash Professional)

Puede utilizar el componente Accordion para presentar formularios de varias partes. Por ejemplo, un componente Accordion con tres elementos secundarios puede presentar formularios para que el usuario escriba su dirección de envío y facturación e información de pago para una transacción de comercio electrónico. Con un Accordion en lugar de varias páginas Web, se minimiza el tráfico en el servidor, lo que permite al usuario optimizar el progreso y la navegabilidad de la aplicación.

Parámetros de Accordion

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Accordion en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

childIcons es una matriz que especifica los identificadores de vinculación de los símbolos de biblioteca que se utilizarán como iconos de los encabezados del componente Accordion. El valor predeterminado es `[]` (matriz vacía).

childLabels es una matriz que especifica las etiquetas de texto que se deben usar en los encabezados del componente Accordion. El valor predeterminado es `[]` (matriz vacía).

childNames es una matriz que especifica los nombres de instancia de los elementos secundarios del componente Accordion. Los valores que escriba serán los nombres de instancia para los símbolos de elementos secundarios que especifique en el parámetro childSymbols. El valor predeterminado es `[]` (matriz vacía).

childSymbols es una matriz que especifica los identificadores de vinculación de los símbolos de biblioteca que se utilizarán para crear los elementos secundarios del componente Accordion. El valor predeterminado es `[]` (matriz vacía).

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente Accordion en el inspector componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no producirá ningún efecto visible.

Puede escribir código ActionScript para controlar otras opciones del componente Accordion utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase Accordion (sólo en Flash Professional)" en la página 49.

Creación de una aplicación con el componente Accordion

En este ejemplo, un desarrollador de aplicaciones está creando la sección de finalización de compra de una tienda en línea. El diseño requiere un componente Accordion con tres formularios donde un usuario introduce una dirección de envío, una dirección de facturación y la información de pago. Los formularios de dirección de envío y dirección de facturación son idénticos.

Para utilizar pantallas con el fin de añadir un componente Accordion a una aplicación:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Aplicación de formularios Flash.
2. Haga doble clic en el texto Form1 e introduzca el nombre **addressForm**.

   Aunque no aparece en la biblioteca, la pantalla addressForm es un símbolo de la clase Screen. Como la clase Screen es una subclase de la clase View, un componente Accordion puede usarla como un elemento secundario.
3. Con el formulario seleccionado, establezca en el inspector de propiedades la propiedad visible del formulario en `false`.

   Esto oculta los contenidos del formulario en la aplicación; el formulario sólo aparece en el componente Accordion.
4. Arrastre componentes como Label y TextInput desde el panel Componentes al formulario para crear un formulario con una dirección ficticia; organícelos y defina sus propiedades en la ficha Parámetros del inspector de componentes.

   Coloque los elementos del formulario en la esquina superior izquierda del formulario. Esta esquina del formulario se coloca en la esquina superior izquierda del componente Accordion.

5. Repita los pasos 2 al 4 para crear una pantalla denominada **checkoutForm**.
6. Cree una nueva pantalla denominada **accordionForm**.
7. Arrastre un componente Accordion desde el panel Componentes al formulario accordionForm y asígnele el nombre **my_acc**.
8. Con my_acc seleccionado, en el inspector de propiedades, siga este procedimiento:
   - Para la propiedad childSymbols, introduzca **addressForm**, **addressForm** y **checkoutForm**.

     Estas cadenas especifican los nombres de las pantallas que se utilizan para crear los elementos secundarios del componente Accordion.

NOTA	Los dos primeros elementos secundarios son instancias de la misma pantalla, ya que el formulario con la dirección de envío y el formulario con la dirección de facturación son idénticos.

   - Para la propiedad childNames, introduzca **shippingAddress**, **billingAddress** y **checkout**.

     Estas cadenas son los nombres ActionScript de los elementos secundarios del componente Accordion.
   - Para la propiedad childLabels, introduzca **Shipping Address**, **Billing Address** y **Checkout**.

     Estas cadenas son las etiquetas de texto de los encabezados del componente Accordion.
9. Seleccione Control > Probar película.

Para añadir un componente Accordion a una aplicación:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Seleccione Insertar > Nuevo símbolo y asígnele el nombre **AddressForm**.
3. En el cuadro de diálogo Crear un nuevo símbolo, haga clic en el botón Avanzado y seleccione Exportar para ActionScript. En el campo Clase de AS 2.0, introduzca **mx.core.View**.

   Para mantener el orden de tabulación en los elementos secundarios del componente Accordion, éstos deben ser también instancias de la clase View.
4. Arrastre componentes como Label y TextInput desde el panel Componentes al escenario para crear un formulario con una dirección ficticia; organícelos y defina sus propiedades en la ficha Parámetros del inspector de componentes.

   Disponga los elementos del formulario con respecto al punto 0,0 (la parte central) del escenario. La coordenada 0,0 del clip de película se coloca en la esquina superior izquierda del componente Accordion.

5. Seleccione Edición > Editar documento para volver a la línea de tiempo principal.
6. Repita los pasos del 2 al 5 para crear un clip de película denominado **CheckoutForm**.
7. Arrastre un componente Accordion desde el panel Componentes para añadirlo al escenario en la línea de tiempo principal.
8. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **my_acc**.
   - Para la propiedad `childSymbols`, introduzca **AddressForm**, **AddressForm** y **CheckoutForm**.

     Estas cadenas especifican los nombres de los clips de película que se utilizan para crear los elementos secundarios del componente Accordion.

NOTA	Los dos primeros elementos secundarios son instancias del mismo clip de película, ya que el formulario con la dirección de envío y el formulario con la dirección de facturación son idénticos.

   - Para la propiedad `childNames`, introduzca **shippingAddress**, **billingAddress** y **checkout**.

     Estas cadenas son los nombres ActionScript de los elementos secundarios del componente Accordion.
   - Para la propiedad `childLabels`, introduzca **Shipping Address**, **Billing Address** y **Checkout**.

     Estas cadenas son las etiquetas de texto de los encabezados del componente Accordion.
   - Para la propiedad `childIcons`, introduzca **AddressIcon**, **AddressIcon** y **CheckoutIcon**.

     Estas cadenas especifican los identificadores de vinculación de los símbolos de clip de película que se utilizan como iconos de los encabezados del componente Accordion. Debe crear estos símbolos de clips de película si desea iconos en los encabezados.
9. Seleccione Control > Probar película.

Para utilizar código ActionScript con el fin de añadir elementos secundarios a un componente Accordion:

1. Seleccione Archivo > Nuevo para crear un documento de Flash.
2. Arrastre un componente Accordion desde el panel Componentes al escenario.
3. En el inspector de propiedades, introduzca el nombre de instancia **my_acc**.

4. Arrastre un componente TextInput a la biblioteca.

   Esto agrega el componente a la biblioteca para que pueda crear dinámicamente una instancia del mismo en el paso 6.

5. En el panel Acciones, en el fotograma 1 de la línea de tiempo, introduzca lo siguiente (este código llama al método `createChild()` para crear sus vistas secundarias):

```
import mx.core.View;

// Crear paneles secundarios para que cada formulario aparezca en el
  objeto my_acc.
my_acc.createChild(View, "shippingAddress", {label: "Shipping
  Address"});
my_acc.createChild(View, "billingAddress", {label: "Billing Address"});
my_acc.createChild(View, "payment", {label: "Payment"});
```

6. En el panel Acciones, en el fotograma 1, bajo el código que se introdujo en el paso 5, introduzca el siguiente código (este código añade dos instancias del componente TextInput a los elementos secundarios del componente Accordion):

```
// Crear entrada de texto secundaria para el panel shippingAddress.
var firstNameChild_obj:Object =
  my_acc.shippingAddress.createChild("TextInput", "firstName", {text:
  "First Name"});

// Establecer la posición de la entrada de texto.
firstNameChild_obj.move(10, 38);
firstNameChild_obj.setSize(110, 20);

// Crear otra entrada de texto secundaria.
var lastNameChild_obj:Object =
  my_acc.shippingAddress.createChild("TextInput", "lastName", {text:
  "Last Name"});

// Establecer la posición de la entrada de texto.
lastNameChild_obj.move(150, 38);
lastNameChild_obj.setSize(140, 20);
```

Personalización del componente Accordion (sólo en Flash Professional)

El componente Accordion puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`).

El método `setSize()` y la herramienta Transformación sólo cambian la anchura de los encabezados del componente Accordion y la anchura y altura de su área de contenido. La altura de los encabezados y la anchura y altura de los elementos secundarios no se ven afectados. Llamar al método `setSize()` es la única forma de cambiar el rectángulo de delimitación de un componente Accordion.

Si los encabezados son demasiado pequeños para contener los textos de etiqueta, éstas se recortan. Si el área de contenido de un componente Accordion es más pequeña que un elemento secundario, éste se recorta.

Utilización de estilos con el componente Accordion

Es posible definir propiedades de estilo para cambiar el aspecto del borde y el fondo de un componente Accordion.

Un componente Accordion utiliza los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "`haloGreen`", "`haloBlue`" y "`haloOrange`".
`backgroundColor`	Ambos	Color del fondo. El color predeterminado es blanco.
`borderStyle`	Ambos	El componente Accordion utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Para más información, consulte “Clase RectBorder” en la página 1103. El valor predeterminado del estilo de borde del componente Accordion es "`solid`".
`headerHeight`	Ambos	Altura de los botones del encabezado, expresada en píxeles. El valor predeterminado es 22.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.

Estilo	Tema	Descripción
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de fuente para las etiquetas del encabezado. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño en puntos para la fuente de las etiquetas del encabezado. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de fuente para las etiquetas de encabezado. Puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de fuente para las etiquetas de encabezado. Puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto. Puede ser `"none"` o `"underline"`.
`openDuration`	Ambos	Duración, en milisegundos, de la animación de transición.
`openEasing`	Ambos	Referencia a una función de interpolación que controla la animación. De forma predeterminada es una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.

Así, por ejemplo, el siguiente código establece como cursiva el aspecto del estilo de la fuente de una instancia de Accordion denominada **my_acc**:

```
my_acc.setStyle("fontStyle", "italic");
```

Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Utilización de aspectos con el componente Accordion

El componente Accordion utiliza aspectos para representar los estados visuales de sus botones de encabezado. Para aplicar un aspecto a los botones y la barra de título durante la edición, modifique los símbolos del aspecto en la carpeta de estados de aspecto Flash UI Components 2/Themes/MMDefault/Accordion Assets de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

Un componente Accordion consta de un borde, fondo, botones de encabezado y elementos secundarios. El borde y el fondo los proporciona la clase RectBorder de manera predeterminada. Para más información sobre la aplicación de aspectos en la clase RectBorder, consulte "Clase RectBorder" en la página 1103. Puede aplicar a los encabezados los aspectos enumerados a continuación.

Propiedad	Descripción	Valor predeterminado
`falseUpSkin`	Estado arriba (normal) del encabezado sobre todos los elementos secundarios contraídos.	`accordionHeaderSkin`
`falseDownSkin`	Estado presionado del encabezado sobre todos los elementos secundarios contraídos.	`accordionHeaderSkin`
`falseOverSkin`	Estado del encabezado sobre todos los elementos secundarios contraídos al desplazar el puntero del ratón sobre él.	`accordionHeaderSkin`
`falseDisabled`	Estado desactivado del encabezado sobre todos los elementos secundarios contraídos.	`accordionHeaderSkin`
`trueUpSkin`	Estado arriba (normal) del encabezado sobre el elemento secundario expandido.	`accordionHeaderSkin`
`trueDownSkin`	Estado presionado del encabezado sobre el elemento secundario expandido.	`accordionHeaderSkin`
`trueOverSkin`	Estado del encabezado sobre el elemento secundario expandido al desplazar el puntero del ratón sobre él.	`accordionHeaderSkin`
`trueDisabledSkin`	Estado desactivado del encabezado sobre el elemento secundario expandido.	`accordionHeaderSkin`

Utilización de ActionScript para dibujar el encabezado del componente Accordion

Los encabezados predeterminados de los temas Halo y Sample usan el mismo elemento de aspecto para todos los estados y dibujan los gráficos mediante ActionScript. La implementación de Halo utiliza una extensión de la clase RectBorder y código personalizado de la API de dibujo para dibujar los estados. En la implementación de Sample se utiliza el mismo aspecto y la misma clase de ActionScript que en el aspecto del componente Button.

Para crear una clase de ActionScript con el fin de usarla como aspecto y proporcionar distintos estados, el aspecto puede leer la propiedad de estilo `borderStyle` para determinar el estado. En la tabla siguiente se muestra el estilo de borde que se establece para cada aspecto:

Propiedad	Estilo del borde
`falseUpSkin`	`falseup`
`falseDownSkin`	`falsedown`
`falseOverSkin`	`falserollover`
`falseDisabled`	`falsedisabled`
`trueUpSkin`	`trueup`
`trueDownSkin`	`truedown`
`trueOverSkin`	`truerollover`
`trueDisabledSkin`	`truedisabled`

Para crear un aspecto personalizado del encabezado de Accordion mediante ActionScript:

1. Cree un nuevo archivo de clase de ActionScript.

   Para este ejemplo, asigne al archivo el nombre **RedGreenBlueHeader.as**.

2. Copie el siguiente código ActionScript al archivo:

```
import mx.skins.RectBorder;
import mx.core.ext.UIObjectExtensions;

class RedGreenBlueHeader extends RectBorder
{
 static var symbolName_str:String = "RedGreenBlueHeader";
 static var symbolOwner_obj:Object = RedGreenBlueHeader;

 function size():Void
 {
  var color_num:Number; // Color
  var borderStyle_str:String = getStyle("borderStyle"); // Atributo de
  Accordion
```

```
        // Definir los colores de cada ficha en el componente Accordion para
        cada estado de ficha.
        switch (borderStyle_str) {
         case "falseup":
         case "falserollover":
         case "falsedisabled":
          color_num = 0x7777FF;
          break;
         case "falsedown":
          color_num = 0x77FF77;
          break;
         case "trueup":
         case "truedown":
         case "truerollover":
         case "truedisabled":
          color_num = 0xFF7777;
          break;
        }

        // Borrar estilo predeterminado y dibujar estilo personalizado.
        clear();
        lineStyle(0, 0, 100);
        beginFill(color_num, 100);
        drawRect(0, 0, __width, __height);
        endFill();
       }

       // requerido para los aspectos
       static function classConstruct():Boolean
       {
        UIObjectExtensions.Extensions();
        _global.skinRegistry["AccordionHeaderSkin"] = true;
        return true;
       }
       static var classConstructed_bl:Boolean = classConstruct();
       static var UIObjectExtensionsDependency_obj:Object =
        UIObjectExtensions;
      }
```

Esta clase crea un cuadrado con el estilo de borde: un cuadro de color azul para los estados seleccionable, puntero del ratón encima y desactivado, un cuadro de color verde para el estado presionado normal y un cuadro de color rojo para el elemento secundario expandido.

3. Guarde el archivo.
4. Cree un archivo FLA nuevo y guárdelo en la misma carpeta que el archivo AS.
5. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
6. Asígnele el nombre `AccordionHeaderSkin`.

7. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
8. Seleccione Exportar para ActionScript.

   El identificador se rellenará automáticamente con `AccordionHeaderSkin`.
9. Establezca la clase de AS 2.0 en `RedGreenBlueHeader`.
10. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
11. En la Escena 1, arrastre un componente Accordion al escenario.
12. Establezca las propiedades del componente Accordion de forma que se muestren varios elementos secundarios.

    Por ejemplo, establezca `childLabels` como una matriz de `[One,Two,Three]` y `childNames` como una matriz de `[one,two,three]`.
13. Seleccione Control > Probar película.

Utilización de clips de película para personalizar el aspecto del encabezado de Accordion

En el ejemplo anterior se ilustra la forma de usar una clase de ActionScript para personalizar el aspecto del encabezado del componente Accordion, que es el método usado por los aspectos proporcionados en los temas Halo y Sample. Sin embargo, como el ejemplo utiliza cuadros coloreados sencillos, en este caso es más sencillo usar distintos símbolos de clip de película como aspectos de encabezado.

Para crear símbolos de clip de película para los aspectos de encabezado del componente Accordion:

1. Cree un nuevo archivo FLA.
2. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
3. Asígnele el nombre `RedAccordionHeaderSkin`.
4. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
5. Seleccione Exportar para ActionScript.

   El identificador se rellenará automáticamente con `RedAccordionHeaderSkin`.
6. Deje el cuadro de texto Clase de AS 2.0 en blanco.
7. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
8. Abra el símbolo nuevo para editarlo.
9. Utilice las herramientas de dibujo para crear un cuadro con relleno en rojo y línea negra.
10. Establezca un estilo de borde muy fino.

11. Defina el cuadro, incluido el borde, para que se posicione en (0,0) y tenga una anchura de 100 y una altura de 100.

    El código ActionScript cambiará el tamaño del aspecto si es necesario.

12. Repita los pasos 2 a 11, cree aspectos de color verde y azul, y asígneles los nombres correspondientes.

13. Haga clic en el botón Atrás para volver a la línea de tiempo principal.

14. Arrastre un componente Accordion al escenario.

15. Establezca las propiedades del componente Accordion de forma que se muestren varios elementos secundarios.

    Por ejemplo, establezca `childLabels` como una matriz de `[One,Two,Three]` y `childNames` como una matriz de `[one,two,three]`.

16. Copie el siguiente código ActionScript al panel Acciones con la instancia de Accordion seleccionada:

```
onClipEvent(initialize) {
  falseUpSkin = "RedAccordionHeaderSkin";
  falseDownSkin = "GreenAccordionHeaderSkin";
  falseOverSkin = "RedAccordionHeaderSkin";
  falseDisabled = "RedAccordionHeaderSkin";
  trueUpSkin = "BlueAccordionHeaderSkin";
  trueDownSkin = "BlueAccordionHeaderSkin";
  trueOverSkin = "BlueAccordionHeaderSkin";
  trueDisabledSkin = "BlueAccordionHeaderSkin";
}
```

17. Seleccione Control > Probar película.

Clase Accordion (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > Accordion

Nombre de clase de ActionScript mx.containers.Accordion

Un componente Accordion contiene elementos secundarios que se muestran de uno en uno. Cada elemento secundario tiene su correspondiente botón de encabezado que se crea cuando se crea el elemento secundario. Un elemento secundario debe ser una instancia de UIObject.

Un símbolo de clip de película se convierte automáticamente en una instancia de la clase UIObject cuando pasa a ser un elemento secundario de un componente Accordion. Sin embargo, para mantener el orden de tabulación en los elementos secundarios del componente Accordion, éstos deben ser también instancias de la clase View. Si utiliza un símbolo de clip de película como elemento secundario, defina el campo Clase de AS 2.0 como mx.core.View para que herede de la clase View.

Si una propiedad de la clase Accordion se define mediante código ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.containers.Accordion.version);
```

NOTA	El código `trace(my_accInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase Accordion

En la tabla siguiente se enumeran los métodos de la clase Accordion.

Método	Descripción
`Accordion.createChild()`	Crea un elemento secundario para una instancia de Accordion.
`Accordion.createSegment()`	Crea un elemento secundario para una instancia de Accordion. Los parámetros para este método son diferentes de los del método `createChild()`.

Método	Descripción
`Accordion.destroyChildAt()`	Elimina un elemento secundario en una posición especificada de índice.
`Accordion.getChildAt()`	Obtiene una referencia a un elemento secundario en una posición especificada de índice.
`Accordion.getHeaderAt()`	Obtiene una referencia a un objeto de encabezado en una posición especificada de índice.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Accordion de la clase UIObject. Al llamar a estos métodos desde el objeto Accordion, debe utilizarse la forma *`accordionInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Accordion de la clase UIComponent. Al llamar a estos métodos desde el objeto Accordion, debe utilizarse la forma *accordionInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase Accordion

En la tabla siguiente se enumeran las propiedades de la clase Accordion.

Propiedad	Descripción
Accordion.numChildren	Número de elementos secundarios de una instancia de Accordion.
Accordion.selectedChild	Referencia al elemento secundario seleccionado.
Accordion.selectedIndex	Posición de índice del elemento secundario seleccionado.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Accordion de la clase UIObject. Al acceder a estas propiedades debe utilizarse la forma *accordionInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
UIObject.height	Sólo lectura; altura del objeto, expresada en píxeles.
UIObject.left	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.right	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Accordion de la clase UIComponent. Al acceder a estas propiedades debe utilizarse la forma *`accordionInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase Accordion

En la tabla siguiente se muestra un evento de la clase Accordion.

Evento	Descripción
`Accordion.change`	Difunde a todos los detectores registrados cuando cambian los valores de las propiedades `selectedIndex` y `selectedChild` de un componente Accordion porque un usuario hace clic en el botón del ratón o pulsa una tecla.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Accordion de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Accordion de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Accordion.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
accordionInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando cambian las propiedades `selectedIndex` y `selectedChild` de un componente Accordion. Este evento sólo se difunde cuando al hacer clic con el ratón o pulsar una tecla se cambia el valor `selectedChild` o `selectedIndex`, no cuando se cambia el valor con ActionScript. Este evento se difunde antes de que se produzca la animación de transición.

Los componentes de la versión 2 utilizan un modelo de distribuidor/detector de eventos. El componente Accordion distribuye un evento `change` cuando se hace clic en uno de sus botones y éste se controla mediante una función (denominada también *controlador*) asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase una referencia al controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

El evento change del componente Accordion también contiene dos propiedades de objetos de evento únicas:

- `newValue` Número; el índice del elemento secundario que se va a seleccionar.
- `prevValue` Número; el índice del elemento secundario que estaba seleccionado anteriormente.

Ejemplo

En el siguiente ejemplo se utiliza una instancia de Accordion denominada `my_acc` que contiene tres paneles secundarios con las etiquetas “Shipping Address”, “Billing Address” y “Payment”. El código define un controlador llamado `my_accListener` y pasa el controlador al método `my_acc.addEventListener()` como el segundo parámetro. El controlador `change` captura el objeto de evento en el parámetro *`eventObject`*. Cuando se difunde el evento `change`, se envía una sentencia `trace` al panel Salida.

```
// Crear un objeto detector nuevo.
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Asignar etiqueta de panel secundario a la variable.
  var selectedChild_str:String = my_acc.selectedChild.label;
  // Llevar a cabo acciones según el elemento secundario seleccionado.
  switch (selectedChild_str) {
  case "Shipping Address":
    trace("One was selected");
    break;
  case "Billing Address":
    trace("Two was selected");
    break;
  case "Payment":
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Accordion.createChild()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
accordionInstance.createChild(classOrSymbolName, instanceName[,
  initialProperties])
```

Parámetros

classOrSymbolName La función constructora para la clase de UIObject de la que se va a crear una instancia o el nombre de vinculación (una referencia al símbolo del que se va a crear una instancia). La clase debe ser UIObject o una subclase de UIObject, pero normalmente es el objeto View o una subclase de View.

instanceName Nombre de la nueva instancia.

initialProperties Parámetro opcional que especifica las propiedades iniciales para la nueva instancia. Puede utilizar las siguientes propiedades:

- `label` Una cadena que especifica la etiqueta de texto que la nueva instancia secundaria utiliza en su encabezado.
- `icon` Una cadena que especifica el identificador de vinculación del símbolo de biblioteca que utiliza el elemento secundario para el icono en su encabezado.

Valor devuelto

Referencia a una instancia del UIObject que es el nuevo elemento secundario creado.

Descripción

Método (heredado de View); crea un elemento secundario para el componente Accordion. Este nuevo elemento secundario se añade al final de la lista de elementos secundarios que pertenecen al componente Accordion. Utilice este método para colocar vistas dentro del componente Accordion. El elemento secundario creado es una instancia de la clase o el símbolo de clip de película especificado en el parámetro *classOrSymbolName*. Puede utilizar las propiedades `label` e `icon` para especificar una etiqueta de texto y un icono para el encabezado del componente Accordion asociado para cada elemento secundario del parámetro *initialProperties*.

Cuando se crea un elemento secundario se le asigna un número de índice en el orden de creación y la propiedad `numChildren` aumenta en una unidad.

Ejemplo

Comience con una instancia de Accordion, denominada `my_acc`, en el escenario. Añada un símbolo a la biblioteca con el identificador de vinculación `payIcon` como icono para el encabezado secundario. El código siguiente crea un elemento secundario denominado `billing` (con la etiqueta "Payment") que es una instancia de la clase View:

```
var child_obj:Object = my_acc.createChild(mx.core.View, "billing", {label:
  "Payment", icon: "payIcon"});
```

El código siguiente también crea un elemento secundario que es una instancia de la clase View, pero utiliza `import` para establecer una referencia al constructor para la clase View:

```
import mx.core.View;
var child_obj:Object = my_acc.createChild(View, "billing", {label:
  "Payment", icon: "payIcon"});
```

O añada un símbolo de clip de película a la biblioteca con el identificador de vinculación `PaymentForm` como elemento secundario de Accordion y el siguiente código creará una instancia de `PaymentForm` denominada `billing` como elemento secundario de `my_acc` (este método es útil cuando se carga contenido dinámico en un símbolo de clip de película y, después, se convierte ese símbolo en un elemento secundario de la instancia de Accordion):

```
var child_obj:Object = my_acc.createChild("PaymentForm", "billing", {label:
  "Payment", icon: "payIcon"});
```

Para ver un ejemplo más complejo, mantenga la instancia de Accordion `my_acc` en el escenario. Después, arrastre un componente Label y un componente TextInput del panel Componentes a la biblioteca del documento actual (de esta forma tendrá un símbolo de TextInput y un símbolo de Label en la biblioteca). Pegue el siguiente código en el primer fotograma de la línea de tiempo principal (reemplazando cualquier código que pueda existir de ejemplos anteriores). El siguiente código crea un elemento secundario que es una instancia de la clase View denominada `billing` y además añade elementos secundarios a `billing` para proporcionar etiquetas y campos de entrada de texto para un formulario:

```
import mx.core.View;
import mx.controls.Label;
import mx.controls.TextInput;
var child_obj:Object = my_acc.createChild(View, "billing",
  {label:"Payment", icon: "payIcon"});
// Crear etiquetas como elementos secundarios de la instancia de View.
var cardType_label:Object = child_obj.createChild(Label, "CardType_label",
  {_x:10, _y:50});
var cardNumber_label:Object = child_obj.createChild(Label,
  "CardNumber_label", {_x:10, _y:100});
// Crear entradas de texto como elementos secundarios de la instancia de
  View.
var cardTypeInput_ti:Object = child_obj.createChild(TextInput,
  "CardType_ti", {_x:150, _y:50});
var cardNumberInput_ti:Object = child_obj.createChild(TextInput,
  "CardNumber_ti", {_x:150, _y:100});
// Rellenar etiquetas.
cardType_label.text = "Card Type";
cardNumber_label.text = "Card Number";
```

Accordion.createSegment()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
accordionInstance.createSegment(classOrSymbolName, instanceName[, label[,
  icon]])
```

Parámetros

classOrSymbolName Una referencia a la función constructora para la clase de UIObject de la que se va a crear una instancia o el nombre de vinculación del símbolo del que se va a crear una instancia. La clase debe ser UIObject o una subclase de UIObject, pero normalmente es el objeto View o una subclase de View.

instanceName Nombre de la nueva instancia.

label Una cadena que especifica la etiqueta de texto que la nueva instancia secundaria utiliza en su encabezado. Este parámetro es opcional.

icon Una cadena de referencia que especifica el identificador de vinculación del símbolo de biblioteca que utiliza el elemento secundario para el icono en su encabezado. Este parámetro es opcional.

Valor devuelto

Referencia a la nueva instancia UIObject creada.

Descripción

Método; crea un elemento secundario para el componente Accordion. Este nuevo elemento secundario se añade al final de la lista de elementos secundarios que pertenecen al componente Accordion. Utilice este método para colocar vistas dentro del componente Accordion. El elemento secundario creado es una instancia de la clase o el símbolo de clip de película especificado en el parámetro *classOrSymbolName*. Puede utilizar los parámetros *label* e *icon* para especificar una etiqueta de texto y un icono para el encabezado del componente Accordion asociado para cada elemento secundario.

El método `createSegment()` difiere del método `createChild()` en que *label* e *icon* se pasan directamente como parámetros, no como propiedades de un parámetro *initalProperties*.

Cuando se crea un elemento secundario, se le asigna un número de índice en el orden de creación y la propiedad `numChildren` aumenta en una unidad.

Ejemplo

Comience con una instancia de Accordion, denominada `my_acc`, en el escenario. Añada un símbolo de clip de película a la biblioteca con el identificador de vinculación `PaymentForm` como elemento secundario de Accordion. A continuación, añada un símbolo a la biblioteca con el identificador de vinculación `payIcon` como icono para el encabezado secundario. En el siguiente ejemplo se crea una instancia del símbolo del clip de película `PaymentForm` denominado `billing` como el último elemento secundario de `my_acc` con la etiqueta de encabezado "Payment" y el icono en la biblioteca:

```
var child_obj:Object = my_acc.createSegment("PaymentForm", "billing",
  "Payment", "payIcon");
```

El código siguiente crea un elemento secundario que es una instancia de la clase View:

```
var child_obj:Object = my_acc.createSegment(mx.core.View, "billing",
  "Payment", "payIcon");
```

El código siguiente también crea un elemento secundario que es una instancia de la clase View, pero utiliza `import` para establecer una referencia al constructor para la clase View:

```
import mx.core.View;
var child_obj:Object = my_acc.createSegment(View, "billing", "Payment",
  "payIcon");
```

Arrastre un componente Label y un componente TextInput del panel Componentes a la biblioteca del documento actual (de modo que tenga tanto un símbolo de TextInput como un símbolo de Label en la biblioteca). El siguiente código crea un elemento secundario que es una instancia de la clase View denominada `billing` y, además, añade elementos secundarios a `billing` para proporcionar etiquetas y campos de entrada de texto para un formulario:

```
import mx.core.View;
import mx.controls.Label;
import mx.controls.TextInput;
var child_obj:Object = my_acc.createSegment(View, "billing", "Payment",
  "payIcon");
// Crear etiquetas como elementos secundarios de la instancia de View.
var cardType_label:Object = child_obj.createChild(Label, "CardType_label",
  {_x:10, _y:50});
var cardNumber_label:Object = child_obj.createChild(Label,
  "CardNumber_label", {_x:10, _y:100});
// Crear entradas de texto como elementos secundarios de la instancia de
  View.
var cardTypeInput_ti:Object = child_obj.createChild(TextInput,
  "CardType_ti", {_x:150, _y:50});
var cardNumberInput_ti:Object = child_obj.createChild(TextInput,
  "CardNumber_ti", {_x:150, _y:100});
// Rellenar etiquetas.
cardType_label.text = "Card Type";
cardNumber_label.text = "Card Number";
```

Accordion.destroyChildAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
accordionInstance.destroyChildAt(index)
```

Parámetros

index Número de índice del elemento secundario del componente Accordion que se debe eliminar. A cada elemento secundario de un componente Accordion se le asigna un número de índice basado en cero en el orden en el que se creó.

Valor devuelto

Ninguno.

Descripción

Método (heredado de View); elimina uno de los elementos secundarios del componente Accordion. El elemento secundario que se va a eliminar se especifica mediante su índice, que se pasa al método en el parámetro *index*. Al llamar a este método se elimina también el encabezado correspondiente.

Si se selecciona un elemento secundario eliminado, se elige un nuevo elemento secundario. Si hay un elemento secundario a continuación, se selecciona dicho elemento. Si no hay ningún elemento secundario a continuación, se selecciona el elemento secundario anterior. Si no hay ningún elemento secundario anterior, la selección no se define.

NOTA

Si se llama a `destroyChildAt()` se disminuye el valor de la propiedad `numChildren` en una unidad.

Ejemplo

El siguiente código elimina el primer elemento secundario de `my_acc` cuando se selecciona el tercer elemento secundario:

```
import mx.core.View;

// Crear paneles secundarios con instancias de la clase View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Crear un objeto detector nuevo.
my_accListener = new Object();
my_accListener.change = function() {
   if ("myMainItem3"){
   my_acc.destroyChildAt(0);
   }
};

my_acc.addEventListener("change", my_accListener);
```

Véase también

`Accordion.createChild()`

Accordion.getChildAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`accordionInstance.getChildAt(`*`index`*`)`

Parámetros

`index` Número de índice de un elemento secundario del componente Accordion. A cada elemento secundario de un componente Accordion se le asigna un índice basado en cero en el orden en el que se creó.

Valor devuelto

Referencia a la instancia de UIObject en el índice especificado.

Descripción

Método; devuelve una referencia al elemento secundario en el índice especificado. A cada elemento secundario del componente Accordion se le asigna un número de índice para su posición. Este número de índice está basado en cero, de modo que el del primer elemento secundario es 0, el del segundo elemento secundario es 1 y así sucesivamente.

Ejemplo

El siguiente código obtiene una referencia al último elemento secundario de `my_acc` y cambia la etiqueta por "Last Child":

```
import mx.core.View;

// Crear paneles secundarios con instancias de la clase View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Obtener referencia para el último objeto secundario.
var lastChild_obj:Object = my_acc.getChildAt(my_acc.numChildren - 1);
// Cambiar la etiqueta del objeto.
lastChild_obj.label = "Last Child";
```

Accordion.getHeaderAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`accordionInstance.getHeaderAt(`*`index`*`)`

Parámetros

`index` Número de índice de un encabezado del componente Accordion. A cada encabezado de un componente Accordion se le asigna un índice basado en cero en el orden en el que se creó.

Valor devuelto

Referencia a la instancia de UIObject en el índice especificado.

Descripción

Método; devuelve una referencia al encabezado en el índice especificado. A cada encabezado del componente Accordion se le asigna un número de índice para su posición. Este número de índice está basado en cero, de modo que el del primer encabezado es 0, el del segundo es 1 y así sucesivamente.

Ejemplo

El siguiente código obtiene una referencia al último encabezado de `my_acc` y muestra la etiqueta en el panel Salida:

```
import mx.core.View;

// Crear paneles secundarios para que cada formulario aparezca en el objeto
  my_acc.
my_acc.createChild(View, "shippingAddress", {label: "Shipping Address"});
my_acc.createChild(View, "billingAddress", {label: "Billing Address"});
my_acc.createChild(View, "payment", {label: "Payment"});

var head3:Object = my_acc.getHeaderAt(2);
trace(head3.label);
```

Accordion.numChildren

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
accordionInstance.numChildren
```

Descripción

Propiedad (heredada de View); indica el número de elementos secundarios (de tipo UIObject) en una instancia de Accordion. Los encabezados no se cuentan como elementos secundarios.

A cada elemento secundario del componente Accordion se le asigna un número de índice para su posición. Este número de índice está basado en cero, de modo que el del primer elemento secundario es 0, el del segundo elemento secundario es 1 y así sucesivamente. El código `my_acc.numChild - 1` siempre hace referencia al último elemento secundario que se ha añadido a un componente Accordion. Por ejemplo, si hay siete elementos secundarios en un componente Accordion, el último tendrá el índice 6. La propiedad `numChildren` no está basada en cero, de modo que el valor de `my_acc.numChildren` será 7. El resultado de 7 - 1 es 6, que es el número de índice del último elemento secundario.

Ejemplo

El siguiente código utiliza `numChildren` para obtener una referencia al último elemento secundario de `my_acc` y cambia la etiqueta por "Last Child":

```
import mx.core.View;

// Crear paneles secundarios con instancias de la clase View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Obtener referencia para el último objeto secundario.
var lastChild_obj:Object = my_acc.getChildAt(my_acc.numChildren - 1);
// Cambiar la etiqueta del objeto.
lastChild_obj.label = "Last Child";
```

Accordion.selectedChild

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
accordionInstance.selectedChild
```

Descripción

Propiedad; el elemento secundario seleccionado (de tipo UIObject) si existen uno o más elementos secundarios; `undefined` si no existe ningún elemento secundario.

Si el componente Accordion tiene elementos secundarios, el código `accordionInstance.selectedChild` es equivalente al código `accordionInstance.getChildAt(accordionInstance.selectedIndex)`.

La definición de esta propiedad en un elemento secundario hace que el componente Accordion inicie la animación de transición para visualizar el elemento secundario especificado.

Si se cambia el valor de `selectedChild`, también cambia el valor de `selectedIndex`.

Si el componente Accordion tiene elementos secundarios, el valor predeterminado es `accordionInstance.getChildAt(0)`. Si el componente Accordion no tiene elementos secundarios, el valor predeterminado es `undefined`.

Ejemplo

En el siguiente ejemplo se detecta cuándo se selecciona un elemento secundario y se muestra el orden de dicho elemento en el panel Salida cada vez que se selecciona un encabezado:

```
// Crear un objeto detector nuevo.
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Asignar etiqueta de panel secundario a la variable.
  var selectedChild_str:String = my_acc.selectedChild.label;
  // Llevar a cabo acciones según el elemento secundario seleccionado.
  switch (selectedChild_str) {
  case "Shipping Address":
    trace("One was selected");
    break;
  case "Billing Address":
    trace("Two was selected");
    break;
  case "Payment":
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Véase también

`Accordion.selectedIndex`

Accordion.selectedIndex

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`accordionInstance.selectedIndex`

Descripción

Propiedad; el índice basado en cero del elemento secundario seleccionado en un componente Accordion con uno o más elementos secundarios. Para un componente Accordion que no tiene vistas secundarias, el único valor válido es `undefined`.

A cada elemento secundario del componente Accordion se le asigna un número de índice para su posición. Este número de índice está basado en cero, de modo que el del primer elemento secundario es 0, el del segundo elemento secundario es 1 y así sucesivamente. Los valores válidos de `selectedIndex` son 0, 1, 2... , *n* - 1, donde *n* es el número de elementos secundarios.

La definición de esta propiedad en un elemento secundario hace que el componente Accordion inicie la animación de transición para visualizar el elemento secundario especificado.

Si se cambia el valor de `selectedIndex` también se cambia el valor de `selectedChild`.

Ejemplo

En el siguiente ejemplo se detecta cuándo se selecciona un elemento secundario y se muestra el orden de dicho elemento en el panel Salida cada vez que se selecciona un encabezado:

```
// Crear un objeto detector nuevo.
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Asignar etiqueta de panel secundario a la variable.
  var selectedChild_num:Number = my_acc.selectedIndex;
  // Llevar a cabo acciones según el elemento secundario seleccionado.
  switch (selectedChild_num) {
  case 0:
    trace("One was selected");
    break;
  case 1:
    trace("Two was selected");
    break;
  case 2:
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Véase también

`Accordion.numChildren`, `Accordion.selectedChild`

CAPÍTULO 3

Componente Alert (sólo en Flash Professional)

3

El componente Alert permite mostrar una ventana con un mensaje para el usuario y botones de respuesta. Esta ventana tiene una barra de título que se puede rellenar con texto, un mensaje que se puede personalizar y botones con etiquetas que se pueden cambiar. Una ventana Alert puede tener cualquier combinación de botones Sí (Yes), No, Aceptar (OK) y Cancelar (Cancel), y puede cambiar las etiquetas de botón mediante las propiedades `Alert.okLabel`, `Alert.yesLabel`, `Alert.noLabel` y `Alert.cancelLabel`. No se puede cambiar el orden de los botones en una ventana Alert; el orden de los botones es siempre Aceptar, Sí, No, Cancelar. Una ventana Alert se cierra cuando un usuario hace clic en cualquiera de los botones de la ventana.

Para mostrar una ventana Alert, llame al método `Alert.show()`. Para llamar al método correctamente, el componente Alert debe estar en la biblioteca. Si arrastra el componente Alert desde el panel Componentes al escenario y después lo elimina, añade el componente a la biblioteca sin que esté visible en el documento.

La previsualización dinámica para el componente Alert es una ventana vacía.

Al añadir un componente Alert a una aplicación, puede utilizar el panel Accesibilidad para hacer que el texto y los botones del componente sean accesibles para los lectores de pantalla. En primer lugar, añada la línea siguiente del código para activar la accesibilidad:

```
mx.accessibility.AlertAccImpl.enableAccessibility();
```

NOTA La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias.

Utilización del componente Alert (sólo en Flash Professional)

Puede utilizar un componente Alert siempre que desee anunciar algo a un usuario. Por ejemplo, puede mostrar una alerta si un usuario no rellena correctamente un formulario, un valor alcanza un precio determinado o si un usuario sale de una aplicación sin guardar la sesión.

Parámetros de Alert

El componente Alert no tiene parámetros de edición. Debe llamar al método `Alert.show()` de ActionScript para mostrar una ventana Alert. Puede utilizar otras propiedades de ActionScript para modificar la ventana Alert de una aplicación. Para más información, consulte "Clase Alert (sólo en Flash Professional)" en la página 74.

Creación de aplicaciones con el componente Alert

El procedimiento siguiente explica cómo añadir un componente Alert a una aplicación durante la edición. En este ejemplo, el componente Alert aparece cuando un valor alcanza un precio determinado.

Para crear una aplicación con el componente Alert:

1. Arrastre el componente Alert desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. En el panel Acciones, especifique el código siguiente en el fotograma 1 de la línea de tiempo para definir un controlador de eventos para el evento `click`:

```
import mx.controls.Alert;

// Definir la acción tras confirmación de alerta.
var myClickHandler:Function = function (evt_obj:Object) {
 if (evt_obj.detail == Alert.OK) {
  trace("start stock app");
 }
};

// Mostrar cuadro de diálogo de alerta.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL, this, myClickHandler, "stockIcon", Alert.OK);
```

Este código crea una ventana Alert con los botones Aceptar y Cancelar. Cuando el usuario hace clic en uno de los botones, Flash llama a la función `myClickHandler`. La función `myClickHandler` indica a Flash que realice un seguimiento de "start stock app" al hacer clic en el botón Aceptar.

NOTA	El método `Alert.show()` incluye un parámetro opcional que muestra un icono en la ventana Alert (en este ejemplo, un icono con el identificador de vinculación "stockIcon"). Para incluir este icono en el ejemplo de prueba, cree un símbolo denominado stockIcon y active Exportar para ActionScript en el cuadro diálogo Propiedades de vinculación o en el cuadro de diálogo Crear un nuevo símbolo. Los gráficos para el símbolo stockIcon deben alinearse en las coordenadas (0,0) del sistema de coordenadas del símbolo.

3. Seleccione Control > Probar película.

Personalización del componente Alert (sólo en Flash Professional)

El componente Alert se sitúa en el centro del componente que se pasó como su parámetro *`parent`*. El elemento principal debe ser un objeto UIComponent. Si es un clip de película, puede registrar el clip como mx.core.View de forma que herede de UIComponent.

La ventana Alert se expande automáticamente de forma horizontal para que quepa el texto del mensaje o cualquiera de los botones visualizados. Si desea visualizar grandes cantidades de texto, incluya saltos de línea en el texto.

El componente Alert no responde al método `setSize()`.

Utilización de estilos con el componente Alert

Es posible definir propiedades de estilo para cambiar el aspecto de un componente Alert. Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente Alert admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Ambos	Color del fondo. El color predeterminado es blanco para el tema Halo y 0xEFEBEF (gris claro) para el tema Sample.
`borderStyle`	Ambos	El componente Alert utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Para más información, consulte "Clase RectBorder" en la página 1103. El componente Alert tiene un valor de `borderStyle` específico de `"alert"` con el tema Halo y `"ouset"` con el tema Sample.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.

Estilo	Tema	Descripción
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.

El componente Alert incluye tres categorías distintas de texto. Si establece las propiedades de texto para el componente Alert, proporciona valores predeterminados para las tres categorías, como se indica a continuación:

```
import mx.controls.Alert;
_global.styles.Alert.setStyle("color", 0x000099);
Alert.show("This is a test alert", "Title");
```

Para establecer los estilos de texto para una categoría individualmente, el componente Alert proporciona propiedades estáticas que son referencias a una instancia de CSSStyleDeclaration.

Propiedad estática	Texto afectado
`buttonStyleDeclaration`	Botón
`messageStyleDeclaration`	Mensaje
`titleStyleDeclaration`	Título

En el ejemplo siguiente se muestra la forma de aplicar cursiva al título de un componente Alert:

```
import mx.controls.Alert;
import mx.styles.CSSStyleDeclaration;

var titleStyles = new CSSStyleDeclaration();
titleStyles.setStyle("fontWeight", "bold");
titleStyles.setStyle("fontStyle", "italic");

Alert.titleStyleDeclaration = titleStyles;

Alert.show("Name is a required field", "Validation Error");
```

Las declaraciones de estilo predeterminadas del título establecen el valor de `fontWeight` en `"bold"`. Si se reemplaza la propiedad `titleStyleDeclaration`, también se reemplaza este valor predeterminado, por lo que se debe establecer explícitamente el valor de `fontWeight` en `"bold"` si se desea usar esa configuración.

NOTA

Los estilos de texto establecidos en un componente Alert proporcionan estilos de texto predeterminados a sus componentes mediante herencia de estilos. Para más información, consulte “Definición de estilos heredados en un contenedor” en *Utilización de componentes*.

Utilización de aspectos con el componente Alert

El componente Alert amplía el componente Window y utiliza su aspecto de fondo para el fondo del título, una instancia de la clase RectBorder para el borde y aspectos del componente Button para los estados visuales de sus botones. Para aplicar un aspecto a los botones y la barra de título durante la edición, modifique los símbolos de Flash UI Components 2/Themes/MMDefault/Window Assets/Elements/TitleBackground y Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*. El borde y el fondo los proporciona la clase RectBorder de manera predeterminada. Para más información sobre la aplicación de aspectos en la clase RectBorder, consulte "Clase RectBorder" en la página 1103.

Un componente Alert utiliza las siguientes propiedades de aspecto para aplicar dinámicamente un aspecto a los botones y a la barra de título.

Propiedad	Descripción	Valor predeterminado
`buttonUp`	Estado sin presionar de los botones.	`ButtonSkin`
`buttonUpEmphasized`	Estado sin presionar del botón predeterminado.	`ButtonSkin`
`buttonDown`	Estado presionado de los botones.	`ButtonSkin`
`buttonDownEmphasized`	Estado presionado del botón predeterminado.	`ButtonSkin`
`buttonOver`	Estado de los botones al desplazar el puntero del ratón sobre ellos.	`ButtonSkin`
`buttonOverEmphasized`	Estado del botón predeterminado al desplazar el puntero del ratón sobre él.	`ButtonSkin`
`titleBackground`	Barra de título de la ventana.	`TitleBackground`

Para definir el título de un componente Alert en un símbolo de clip de película personalizado:

1. Cree un nuevo archivo FLA.
2. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
3. Asígnele el nombre `TitleBackground`.
4. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
5. Seleccione Exportar para ActionScript.
6. El identificador se rellenará automáticamente con `TitleBackground`.

7. Establezca `mx.skins.SkinElement` como clase de AS 2.0.

   SkinElement es una clase simple que puede utilizarse para todos los elementos de aspecto que no proporcionan su propia implementación de ActionScript. Proporciona el movimiento y las funciones de cambio de tamaño que requiere el marco de componentes de la versión 2 de la arquitectura de componentes de Macromedia.
8. Verifique que Exportar en primer fotograma esté seleccionado.
9. Haga clic en Aceptar.
10. Abra el símbolo nuevo para editarlo.
11. Utilice las herramientas de dibujo para crear un cuadro con relleno en rojo y línea negra.
12. Establezca un estilo de borde muy fino.
13. Defina el cuadro, incluido el borde, de forma que se posicione en (0,0) y tenga una anchura de 100 y una altura de 22.

    El componente Alert define la anchura adecuada del aspecto, pero utiliza la altura existente como altura del título.
14. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
15. Arrastre un componente Alert al escenario y elimínelo.

    Con esta acción se añade el componente Alert a la biblioteca y está disponible en tiempo de ejecución.
16. Añada código ActionScript a la línea de tiempo principal para crear una instancia de Alert de ejemplo.

```
import mx.controls.Alert;
Alert.show("This is a skinned Alert component","Title");
```

17. Seleccione Control > Probar película.

Clase Alert (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > Componente Window > Alert

Nombre de clase de ActionScript mx.controls.Alert

Para utilizar el componente Alert, se debe arrastrar un componente Alert al escenario y eliminarlo, de modo que el componente esté en la biblioteca del documento pero no visible en la aplicación. A continuación, se llama a `Alert.show()` para abrir una ventana Alert. Puede pasar a `Alert.show()` parámetros que añaden un mensaje, una barra de título y botones a la ventana Alert.

Puesto que ActionScript es asíncrono, el componente Alert no produce bloqueos, de modo que las líneas de código ActionScript después de la llamada a `Alert.show()` se ejecutan inmediatamente. Debe añadir detectores para controlar los eventos `click` que se difunden cuando un usuario hace clic en un botón y, a continuación, seguir con el código después de que se haya difundido el evento.

NOTA	En entornos operativos donde se produzcan bloqueos (por ejemplo, Microsoft Windows), no se devolverá una llamada a `Alert.show()` hasta que el usuario haya llevado a cabo una acción, como hacer clic en un botón.

Para más información sobre la clase Alert, consulte "Componente Window" en la página 1507 y "Clase PopUpManager" en la página 1025.

Resumen de métodos de la clase Alert

En la tabla siguiente se muestra el método de la clase Alert.

Método	Descripción
`Alert.show()`	Crea una ventana Alert con parámetros opcionales.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Alert de la clase UIObject.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.

Método	Descripción
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Alert de la clase UIComponent.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Métodos heredados de la clase Window

En la tabla siguiente se enumeran los métodos que hereda la clase Alert de la clase Window.

Método	Descripción
Window.deletePopUp()	Elimina una instancia de Window creada por PopUpManager.createPopUp().

Resumen de propiedades de la clase Alert

En la tabla siguiente se enumeran las propiedades de la clase Alert.

Propiedad	Descripción
`Alert.buttonHeight`	Altura de cada botón, expresada en píxeles. El valor predeterminado es 22.
`Alert.buttonWidth`	Anchura de cada botón, expresada en píxeles. El valor predeterminado es 100.
`Alert.CANCEL`	Valor hexadecimal constante que indica si se debe mostrar un botón Cancelar en la ventana Alert.
`Alert.cancelLabel`	Texto de la etiqueta del botón Cancelar.
`Alert.NO`	Valor hexadecimal constante que indica si se debe mostrar un botón No en la ventana Alert.
`Alert.noLabel`	Texto de la etiqueta del botón No.
`Alert.OK`	Valor hexadecimal constante que indica si se debe mostrar un botón Aceptar en la ventana Alert.
`Alert.okLabel`	Texto de la etiqueta del botón Aceptar.
`Alert.YES`	Valor hexadecimal constante que indica si se debe mostrar un botón Sí en la ventana Alert.
`Alert.yesLabel`	Texto de la etiqueta del botón Sí.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Alert de la clase UIObject. Al llamar a estas propiedades desde el objeto Alert, debe utilizarse la forma `Alert.`*`propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Alert de la clase UIComponent. Al llamar a estas propiedades desde el objeto Alert, debe utilizarse la forma `Alert.`*`propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase Window

En la tabla siguiente se enumeran las propiedades que hereda la clase Alert de la clase Window.

Propiedad	Descripción
`Window.closeButton`	Indica si la barra de título incluye un botón de cierre (`true`) o no (`false`).
`Window.content`	Referencia al contenido (clip de película raíz) de la ventana.
`Window.contentPath`	Define el nombre del contenido que ha de aparecer en la ventana.
`Window.title`	Texto que aparece en la barra de título.
`Window.titleStyleDeclaration`	Declaración de estilos que asigna formato al texto de la barra de título.

Resumen de eventos de la clase Alert

En la tabla siguiente se muestra un evento de la clase Alert.

Evento	Descripción
`Alert.click`	Se difunde cuando se hace clic en un botón de la ventana Alert.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Alert de la clase UIObject. Al llamar a estos eventos desde el objeto Alert, debe utilizarse la forma `Alert.eventName`.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Alert de la clase UIComponent. Al llamar a estos eventos desde el objeto Alert, debe utilizarse la forma `Alert.eventName`.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase Window

En la tabla siguiente se enumeran los eventos que hereda la clase Alert de la clase Window.

Evento	Descripción
`Window.click`	Se difunde cuando se hace clic en (se suelta) el botón de cierre.
`Window.complete`	Se difunde cuando se crea una ventana.

Alert.buttonHeight

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.buttonHeight
```

Descripción

Propiedad (clase); una propiedad de clase (estática) que cambia la altura de los botones. El valor predeterminado es 22.

Ejemplo

Con un componente Alert en la biblioteca, este ejemplo cambia el tamaño de los botones:

```
import mx.controls.Alert;

// Ajustar tamaños de botón.
Alert.buttonHeight = 50;
Alert.buttonWidth = 150;

// Mostrar cuadro de diálogo de alerta.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL);
```

Véase también

`Alert.buttonWidth`

Alert.buttonWidth

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.buttonWidth
```

Descripción

Propiedad (clase); una propiedad (estática) de clase que cambia la anchura de los botones. El valor predeterminado es 100.

Ejemplo

Con un componente Alert en la biblioteca, añada este código ActionScript al primer fotograma de la línea de tiempo principal para cambiar el tamaño de los botones:

```
import mx.controls.Alert;

// Ajustar tamaños de botón.
Alert.buttonHeight = 50;
Alert.buttonWidth = 150;

// Mostrar cuadro de diálogo de alerta.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL);
```

Véase también

`Alert.buttonHeight`

Alert.CANCEL

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.CANCEL
```

Descripción

Propiedad (constante); una propiedad con el valor hexadecimal constante 0x8. Esta propiedad se puede usar para el parámetro *flags* o *defaultButton* del método `Alert.show()`. Cuando se usa como un valor para el parámetro *flags*, esta propiedad indica que se debe mostrar un botón Cancelar en la ventana Alert. Cuando se usa como un valor del parámetro *defaultButton*, el botón Cancelar tiene la selección inicial y se activa cuando el usuario presiona Intro (Windows) o Retorno (Macintosh). Si el usuario presiona el tabulador para llegar a otro botón, se activará dicho botón cuando el usuario presione Intro.

Ejemplo

En el ejemplo siguiente se usa `Alert.CANCEL` y `Alert.OK` como valores para el parámetro *flags* y se muestra un componente Alert con un botón Aceptar y un botón Cancelar:

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.OK |
  Alert.CANCEL, this);
```

Alert.cancelLabel

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.cancelLabel
```

Descripción

Propiedad (clase); una propiedad (estática) de clase que indica el texto de la etiqueta del botón Cancelar.

Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón Cancel como “cancellation”:

```
Alert.cancelLabel = "cancellation";
```

Alert.click

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
var clickHandler:Object = function(eventObject:Object) {
  // Introducir aquí el código.
}
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,
  defaultButton]]]]]])
```

Descripción

Evento; se difunde al detector registrado cuando se hace clic en el botón Aceptar, Sí, No o Cancelar.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. El componente Alert distribuye un evento `click` cuando se hace clic en uno de sus botones y el evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `Alert.show()` y pase el nombre del controlador como parámetro. Cuando se hace clic en un botón de la ventana Alert, se llamará al detector.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto de evento del evento `Alert.click` tiene una propiedad `detail` adicional cuyo valor es `Alert.OK`, `Alert.CANCEL`, `Alert.YES` o `Alert.NO`, en función del botón en el que se hizo clic. Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

Con un componente Alert en la biblioteca, añada este código ActionScript al primer fotograma de la línea de tiempo principal para crear un controlador de eventos denominado `myClickHandler`: El controlador de eventos se pasa al método `Alert.show()` como el quinto parámetro. El objeto de evento se captura mediante el controlador `myClickHandler` en el parámetro `evt`. A continuación, la propiedad `detail` del objeto de evento se utiliza dentro de una sentencia `trace` para enviar el nombre del botón en el que se hizo clic (`Alert.OK` o `Alert.CANCEL`) al panel Salida, como se indica a continuación:

```
import mx.controls.Alert;

// Definir acciones de botón.
var myClickHandler:Function = function (evt_obj:Object) {
 switch (evt_obj.detail) {
 case Alert.OK :
  trace("You clicked: " + Alert.okLabel);
  break;
 case Alert.CANCEL :
  trace("You clicked: " + Alert.cancelLabel);
  break;
 }
};

// Mostrar cuadro de diálogo.
Alert.show("This is a test of errors", "Error", Alert.OK | Alert.CANCEL,
  this, myClickHandler);
```

Alert.NO

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.NO
```

Descripción

Propiedad (constante); una propiedad con el valor hexadecimal constante 0x2. Esta propiedad se puede usar para el parámetro *flags* o *defaultButton* del método `Alert.show()`. Cuando se usa como un valor para el parámetro *flags*, esta propiedad indica que se debe mostrar un botón No en la ventana Alert. Cuando se usa como un valor del parámetro *defaultButton*, el botón Cancelar tiene la selección inicial y se activa cuando el usuario presiona Intro (Windows) o Retorno (Macintosh). Si el usuario presiona el tabulador para llegar a otro botón, se activará dicho botón cuando el usuario presione Intro.

Ejemplo

En el ejemplo siguiente se usa `Alert.NO` y `Alert.YES` como valores para el parámetro *flags* y se muestra un componente Alert con un botón No y un botón Sí:

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.NO |
  Alert.YES, this);
```

Alert.noLabel

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.noLabel
```

Descripción

Propiedad (clase); una propiedad (estática) de clase que indica el texto de la etiqueta del botón No.

Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón No como “nyet”:

```
Alert.noLabel = "nyet";
```

Alert.NONMODAL

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.NONMODAL
```

Descripción

Propiedad (constante); una propiedad con el valor hexadecimal constante 0x8000. Esta propiedad se puede usar para el parámetro *flags* del método `Alert.show()`. Esta propiedad indica que una ventana Alert debe ser amodal, lo que permite a los usuarios interactuar con los botones y las instancias situadas por debajo de la ventana que se muestra. De forma predeterminada, las ventanas generadas con `Alert.show()` son modales, lo que significa que los usuarios no pueden hacer clic en nada que no sea la ventana que actualmente está abierta.

Ejemplo

En el siguiente ejemplo se muestran dos instancias del componente Button en el escenario. Si se hace clic en un botón, se abre una ventana modal que impide al usuario hacer clic en más botones hasta que no se cierre la ventana Alert. El segundo botón abre una ventana amodal, lo que permite al usuario seguir haciendo clic en los botones que se encuentran por debajo de la ventana Alert amodal abierta en ese momento. Para probar este ejemplo, añada instancias del componente Alert y del componente Button a la biblioteca del documento actual y añada el siguiente código al fotograma 1 de la línea de tiempo principal:

```
import mx.controls.Alert;

this.createClassObject(mx.controls.Button, "modal_button", 10, {_x:10,
  _y:10});
this.createClassObject(mx.controls.Button, "nonmodal_button", 20, {_x:120,
  _y:10});

modal_button.label = "modal";
modal_button.addEventListener("click", modalListener);
function modalListener(evt_obj:Object):Void {
  var a:Alert = Alert.show("This is a modal Alert window", "Alert Test",
  Alert.OK, this);
  a.move(100, 100);
}
```

```
nonmodal_button.label = "nonmodal";
nonmodal_button.addEventListener("click", nonmodalListener);
function nonmodalListener(evt_obj:Object):Void {
  var a:MovieClip = Alert.show("This is a nonmodal Alert window", "Alert
  Test", Alert.OK | Alert.NONMODAL, this);
  a.move(100, 100);
}
```

Alert.OK

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.OK
```

Descripción

Propiedad (constante); una propiedad con el valor hexadecimal constante 0x4. Esta propiedad se puede usar para el parámetro *flags* o *defaultButton* del método `Alert.show()`. Cuando se usa como un valor para el parámetro *flags*, esta propiedad indica que se debe mostrar un botón Aceptar en la ventana Alert. Cuando se usa como un valor del parámetro *defaultButton*, el botón Aceptar tiene la selección inicial y se activa cuando el usuario presiona Intro (Windows) o Retorno (Macintosh). Si el usuario presiona el tabulador para llegar a otro botón, se activará dicho botón cuando el usuario presione Intro.

Ejemplo

En el ejemplo siguiente se usa `Alert.OK` y `Alert.CANCEL` como valores para el parámetro *flags* y se muestra un componente Alert con un botón Aceptar y un botón Cancelar:

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.OK |
  Alert.CANCEL, this);
```

Alert.okLabel

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.okLabel
```

Descripción

Propiedad (clase); una propiedad (estática) de clase que indica el texto de la etiqueta del botón Aceptar.

Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón OK como “okay”:

```
Alert.okLabel = "okay";
```

Alert.show()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,
  defaultButton]]]]]])
```

Parámetros

message Mensaje que se mostrará.

title Texto de la barra de título del componente Alert. Este parámetro es opcional; si lo omite, la barra de título estará vacía.

flags Parámetro opcional que indica los botones que se mostrarán en la ventana Alert. El valor predeterminado es `Alert.OK`, que muestra un botón Aceptar. Cuando utilice más de un valor, separe los valores con el carácter `|`. Especifique uno o varios de los siguientes valores: `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`.

También puede utilizar `Alert.NONMODAL` para indicar que la ventana Alert no es modal. Una ventana que no es modal permite al usuario interactuar con otras ventanas de la aplicación.

`parent` Ventana principal del componente Alert. La ventana Alert se centra en la ventana principal. Utilice el valor `null` o `undefined` para especificar la línea de tiempo _root. La ventana principal debe ser una subclase de la clase UIComponent, ya sea otro componente de Flash que sea una subclase de UIComponent, o una ventana personalizada que sea una subclase de UIComponent (para más información, consulte "Herencia" en *Aprendizaje de ActionScript 2.0 en Flash*. Este parámetro es opcional.

`clickHandler` Controlador para los eventos `click` difundidos cuando se hace clic en los botones. Además de las propiedades del objeto del evento click estándar hay una propiedad `detail`, que contiene el valor del indicador de botón en el que se ha hecho clic (`Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`). Este controlador puede ser una función o un objeto. Para más información, consulte "Utilización de detectores para gestionar eventos" en *Utilización de componentes*.

`icon` Una cadena que es el identificador de vinculación de un símbolo en la biblioteca; este símbolo se usa como un icono mostrado a la izquierda del texto de alerta. Este parámetro es opcional.

`defaultButton` Indica el botón que tiene la selección inicial y en el que se hace clic cuando un usuario presiona Intro (Windows) o Retorno (Macintosh). Si un usuario se desplaza a otro botón mediante el tabulador, se activará dicho botón cuando se presione la tecla Intro.

Este parámetro puede tener uno de estos valores: `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`.

Valor devuelto

La instancia de Alert creada.

Descripción

Método (clase); método de clase (estática) que muestra una ventana Alert con un mensaje, un título opcional, botones opcionales y un icono opcional. El título de la alerta aparece en la parte superior de la ventana y se alinea a la izquierda. El icono aparece a la izquierda del texto del mensaje. Los botones aparecen centrados debajo del texto del mensaje y el icono.

Ejemplo

El código siguiente es un sencillo ejemplo de una ventana Alert modal con un botón Aceptar:

```
mx.controls.Alert.show("Hello, world!");
```

En el código siguiente, se define un controlador de acciones del ratón que envía un mensaje al panel Salida en el que se indica el botón en el que se ha hecho clic. (Es necesario tener un componente Alert en la biblioteca para que este código muestre una alerta; para añadir el componente a la biblioteca, arrástrelo al escenario y, a continuación, bórrelo):

```
import mx.controls.Alert;

// Definir acciones de botón.
var myClickHandler:Function = function (evt_obj:Object) {
 if (evt_obj.detail == Alert.OK) {
  trace(Alert.okLabel);
 } else if (evt_obj.detail == Alert.CANCEL) {
  trace(Alert.cancelLabel);
 }
};

// Mostrar cuadro de diálogo.
var dialog_obj:Object = Alert.show("Test Alert", "Test", Alert.OK |
  Alert.CANCEL, null, myClickHandler, "testIcon", Alert.OK);
```

Alert.YES

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.YES
```

Descripción

Propiedad (constante); una propiedad con el valor hexadecimal constante 0x1. Esta propiedad se puede usar para el parámetro *flags* o *defaultButton* del método `Alert.show()`. Cuando se usa como un valor para el parámetro *flags*, esta propiedad indica que se debe mostrar un botón Sí en la ventana Alert. Cuando se usa como un valor del parámetro *defaultButton*, el botón Sí tiene la selección inicial y se activa cuando el usuario presiona Intro (Windows) o Retorno (Macintosh). Si el usuario presiona el tabulador para llegar a otro botón, se activará dicho botón cuando el usuario presione Intro.

Ejemplo

En el ejemplo siguiente se usa `Alert.NO` y `Alert.YES` como valores para el parámetro *`flags`* y se muestra un componente Alert con un botón No y un botón Sí:

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.NO |
  Alert.YES, this);
```

Alert.yesLabel

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
Alert.yesLabel
```

Descripción

Propiedad (clase); una propiedad (estática) de clase que indica el texto de la etiqueta del botón Sí.

Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón OK como “da”:

```
Alert.yesLabel = "da";
```

CAPÍTULO 4

Componente Button

4

El componente Button es un botón rectangular de la interfaz de usuario que puede cambiarse de tamaño. También se le puede añadir un icono personalizado y cambiar su comportamiento de botón de comando a conmutador. El conmutador permanece presionado cuando se hace clic en él y recupera su estado original al repetirse el clic.

Un botón puede estar activado o desactivado en una aplicación. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado. Si está activado, el botón se selecciona cuando el usuario hace clic en él o usa el tabulador para llegar hasta él. Cuando una instancia de Button está seleccionada, es posible utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Barra espaciadora	Presiona o libera el componente y activa el evento `click`.
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia de Button refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes. Sin embargo, en la previsualización dinámica los iconos personalizados se representan en el escenario con un cuadrado gris.

Cuando se añade el componente Button a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la siguiente línea de código:

```
mx.accessibility.ButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias.

Utilización del componente Button

Un botón es una parte fundamental de cualquier formulario o aplicación Web. Utilice botones siempre que necesite que el usuario inicie un evento. Por ejemplo, la mayoría de los formularios contienen un botón Enviar. También puede añadir botones Anterior y Siguiente a una presentación.

Para añadir un icono a un botón, es necesario seleccionar o crear un clip de película o un símbolo gráfico para utilizarlo como icono. El símbolo debe registrarse en la posición 0,0 para que el botón tenga el diseño adecuado. Seleccione el símbolo del icono en el panel Biblioteca, abra el cuadro de diálogo Vinculación desde el menú emergente Biblioteca e introduzca un identificador de vinculación. Éste es el valor que debe introducir como parámetro icon en el inspector de propiedades o el inspector de componentes. También puede introducirlo en la propiedad `Button.icon` de ActionScript.

NOTA	Si el icono es más grande que el botón, sobresaldrá por los bordes de éste.

Para designar un botón como el botón de comando predeterminado de una aplicación (el botón que recibe el evento click cuando un usuario presiona Intro), use `FocusManager.defaultPushButton`.

Parámetros de Button

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Button en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

icon añade un icono personalizado al botón. El valor es el identificador de vinculación de un clip de película o símbolo gráfico de la biblioteca; no hay valor predeterminado.

label define el valor del texto del botón; el valor predeterminado es Button.

labelPlacement orienta el texto de la etiqueta del botón con respecto al icono. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom`; el valor predeterminado es `right`. Para más información, consulte `Button.labelPlacement`.

selected si el valor del parámetro toggle es `true`, este parámetro especifica si el botón está presionado (`true`) o no (`false`). El valor predeterminado es`false`.

toggle convierte el botón en un conmutador. Si el valor es `true`, el botón permanece en el estado presionado cuando se hace clic en él y recupera el estado sin presionar cuando se vuelve a hacer clic en él. Si el valor es `false`, el botón se comporta como un botón de comando normal. El valor predeterminado es `false`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente Button en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no producirá ningún efecto visible.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente Button utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase Button" en la página 104.

Creación de aplicaciones con el componente Button

El siguiente procedimiento explica cómo añadir un componente Button a una aplicación durante la edición. En este ejemplo, el botón muestra un mensaje al hacer clic en él.

Para crear una aplicación con el componente Button:

1. Arrastre un componente Button desde el panel Componentes al escenario.
2. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **my_button**.
   - Introduzca **Click me** para el parámetro label.
   - Introduzca **BtnIcon** en el parámetro icon.

     Para utilizar un icono, en la biblioteca debe haber un clip de película o un símbolo gráfico con un identificador de vinculación para emplearlo como parámetro icon. En este ejemplo, el identificador de vinculación es BtnIcon.
   - Establezca la propiedad `toggle` en `true`.
3. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
function clicked(){
  trace("You clicked the button!");
}
my_button.addEventListener("click", clicked);
```

La última línea de código llama a una función de controlador de eventos `clicked` para el evento "click". Ésta utiliza el método "EventDispatcher.addEventListener()" en la página 516 con una función personalizada para controlar el evento.

4. Seleccione Control > Probar película.
5. Al hacer clic en el botón, Flash muestra un mensaje que indica que ha hecho clic en el botón.

Para crear un botón mediante código ActionScript:

1. Arrastre el componente Button desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones para crear una instancia de Button:

```
this.createClassObject(mx.controls.Button, "my_button", 10,
  {label:"Click me"});
my_button.move(20, 20);
```

   Se utiliza el método `UIObject.createClassObject()` para crear la instancia de Button denominada **my_button** y especificar una propiedad de etiqueta. A continuación, el código utiliza el método `UIObject.move()` para colocar el botón.

3. Añada el siguiente código ActionScript para crear un detector de eventos y una función de controlador de eventos:

```
function clicked() {
  trace("You clicked the button!");
}
my_button.addEventListener("click", clicked);
```

   Se utiliza el método "EventDispatcher.addEventListener()" en la página 516 con una función personalizada para controlar el evento.

4. Seleccione Control > Probar película.
5. Al hacer clic en el botón, Flash muestra un mensaje que indica que ha hecho clic en el botón.

Mientras utiliza el componente Button con otros componentes, puede crear funciones de controlador de eventos más sofisticadas. En este ejemplo, el evento "click" hace que el componente Accordion cambie la visualización de los paneles.

Para utilizar un evento Button con otro componente:

1. Arrastre el componente Button desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. Arrastre el componente Accordion desde el panel Componentes a la biblioteca del documento actual.
3. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones para crear una instancia de Button:

```
this.createClassObject(mx.containers.Accordion, "my_acc", 0);
my_acc.move(10, 40);
my_acc.createChild(mx.core.View, "panelOne", {label: "Panel One"});
my_acc.createChild(mx.core.View, "panelTwo", {label: "Panel Two"});

this.createClassObject(mx.controls.Button, "panelOne_button", 10,
  {label:"Panel One"});
panelOne_button.move(10, 10);
this.createClassObject(mx.controls.Button, "panelTwo_button", 20,
  {label:"Panel Two"});
panelTwo_button.move(120, 10);
```

   En este proceso se utiliza el método `UIObject.createClassObject()` para crear las instancias de Button y Accordion. A continuación, el código utiliza el método `UIObject.move()` para colocar las instancias.
4. Añada el siguiente código ActionScript para crear detectores de eventos y funciones de controlador de eventos:

```
// Crear controlador para el evento de botón.
function changePanel(evt_obj:Object):Void {
 // Cambiar la vista de Accordion según el botón seleccionado.
 switch (evt_obj.target._name) {
 case "panelOne_button" :
  my_acc.selectedIndex = 0;
  break;
 case "panelTwo_button" :
  my_acc.selectedIndex = 1;
  break;
 }
}

// Añadir detectores para los botones.
panelOne_button.addEventListener("click", changePanel);
panelTwo_button.addEventListener("click", changePanel);
```

   En este proceso se utiliza el método `EventDispatcher.addEventListener()` con una función personalizada para controlar los eventos.
5. Seleccione Control > Probar película.
6. Al hacer clic en un botón, el componente Accordion cambia el panel actual.

Personalización del componente Button

El componente Button puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o cualquier método o propiedad aplicable de la clase Button (véase "Clase Button" en la página 104). Cuando se modifica el tamaño del botón, no cambia el tamaño del icono ni de la etiqueta.

El recuadro de delimitación de una instancia de Button es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

Si el icono es más grande que el botón, el icono sobresaldrá por los bordes del botón.

Utilización de estilos con el componente Button

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de Button. Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente Button admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Sample	Color del fondo. El valor predeterminado es 0xEFEBEF (gris claro). El tema Halo utiliza 0xF8F8F8 (gris muy claro) como color de fondo del botón cuando el botón no está presionado y `themeColor` cuando el botón está presionado. Sólo puede modificar el color de fondo del estado sin presionar en el tema Halo aplicando un aspecto al botón. Véase "Utilización de aspectos con el componente Button" en la página 98.

Estilo	Tema	Descripción
`borderStyle`	Sample	El componente Button utiliza una instancia de RectBorder como borde en el tema Sample y responde a los estilos definidos en esa clase. Véase "Clase RectBorder" en la página 1103. Con el tema Halo, el componente Button utiliza un borde redondeado personalizado cuyos colores no se pueden modificar (salvo `themeColor`).
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

Utilización de aspectos con el componente Button

El componente Button incluye 32 aspectos diferentes que se pueden personalizar para que se correspondan con 16 estados distintos del borde y el icono. Para aplicar un aspecto al componente Button durante la edición, cree nuevos símbolos de clip de película con los gráficos deseados y establezca los identificadores de vinculación de símbolo mediante código ActionScript. Para más información, consulte "Utilización de ActionScript para dibujar aspectos del componente Button" en la página 100.

La implementación predeterminada de los aspectos del componente Button proporcionada con los temas Halo y Sample utiliza la API de dibujo de ActionScript para dibujar los estados de botón, y utiliza un símbolo de clip de película individual asociado con una clase de ActionScript para proporcionar todos los aspectos del componente Button.

El componente Button tiene muchos aspectos porque un botón tiene muchos estados distintos, además de un borde y un icono distinto para cada estado. El estado de una instancia de Button se controla mediante cuatro propiedades y la interacción con el usuario. Las siguientes propiedades afectan a los aspectos:

Propiedad	Descripción
`emphasized`	Proporciona dos apariencias distintas para instancias de Button y se suele usar para resaltar un botón, como el botón predeterminado de un formulario.
`enabled`	Indica si el botón permite la interacción con el usuario.
`toggle`	Proporciona un valor de estado seleccionado y de estado no seleccionado y usa aspectos diferentes para indicar el valor actual. Para una instancia de Button cuya propiedad `toggle` esté establecida en `false`, se usarán los aspectos `false`. Cuando el valor de la propiedad `toggle` es `true`, el aspecto dependerá de la propiedad `selected`.
`selected`	Cuando se establece la propiedad `toggle` en `true`, determina si el componente Button está seleccionado (`true` o `false`). Para identificar el valor se usan distintos aspectos, que son de forma predeterminada la única manera de indicar este valor en pantalla.

Si un botón está activado, muestra el estado correspondiente al desplazamiento del puntero sobre él. Cuando se presiona el botón, recibe la selección de entrada y muestra el estado presionado. Cuando se suelta el botón del ratón, el botón recupera el estado correspondiente al desplazamiento del puntero sobre él. Si el puntero se retira del botón estando el ratón presionado, el botón recupera su estado original y conserva la selección de entrada. Si el parámetro toggle está establecido en `true`, el estado del botón no cambiará hasta que, estando el puntero del ratón sobre el botón, se suelte el botón del ratón.

Si el botón está desactivado, muestra el estado desactivado sea cual sea la acción del usuario.

El componente Button admite las siguientes propiedades de aspecto:

Propiedad	Descripción
`falseUpSkin`	Estado sin presionar (normal).
`falseDownSkin`	Estado presionado.
`falseOverSkin`	Estado cuando se desplaza el puntero sobre él.
`falseDisabledSkin`	Estado desactivado.
`trueUpSkin`	Estado conmutado.
`trueDownSkin`	Estado presionado conmutado.
`trueOverSkin`	Estado conmutado cuando se desplaza el puntero sobre él.
`trueDisabledSkin`	Estado desactivado conmutado.
`falseUpSkinEmphasized`	Estado sin presionar (normal) de un botón resaltado.
`falseDownSkinEmphasized`	Estado presionado de un botón resaltado.
`falseOverSkinEmphasized`	Estado de un botón resaltado cuando se desplaza el puntero sobre él.
`falseDisabledSkinEmphasized`	Estado desactivado de un botón resaltado.
`trueUpSkinEmphasized`	Estado conmutado de un botón resaltado.
`trueDownSkinEmphasized`	Estado presionado conmutado de un botón resaltado.
`trueOverSkinEmphasized`	Estado conmutado de un botón resaltado, cuando se desplaza el puntero sobre él.
`trueDisabledSkinEmphasized`	Estado desactivado conmutado de un botón resaltado.
`falseUpIcon`	Estado sin presionar del icono.
`falseDownIcon`	Estado presionado del icono.
`falseOverIcon`	Estado del icono cuando se desplaza el puntero sobre él.
`falseDisabledIcon`	Estado desactivado del icono.
`trueUpIcon`	Estado conmutado del icono.
`trueOverIcon`	Estado conmutado del icono cuando se desplaza el puntero sobre él.
`trueDownIcon`	Estado presionado conmutado del icono.
`trueDisabledIcon`	Estado desactivado conmutado del icono.
`falseUpIconEmphasized`	Estado sin presionar del icono de un botón resaltado.
`falseDownIconEmphasized`	Estado presionado del icono de un botón resaltado.

Propiedad	Descripción
`falseOverIconEmphasized`	Estado del icono de un botón resaltado cuando se desplaza el puntero sobre él
`falseDisabledIconEmphasized`	Estado desactivado del icono de un botón resaltado.
`trueUpIconEmphasized`	Estado conmutado del icono de un botón resaltado.
`trueOverIconEmphasized`	Estado conmutado del icono de un botón resaltado cuando se desplaza el puntero sobre él.
`trueDownIconEmphasized`	Estado presionado conmutado del icono de un botón resaltado.
`trueDisabledIconEmphasized`	Estado desactivado conmutado del icono de un botón resaltado.

El valor predeterminado de todas las propiedades del aspecto cuyo nombre acabe en "Skin" es `ButtonSkin` y el valor predeterminado de todas las propiedades "Icon" es `undefined`. Las propiedades con el sufijo "Skin" proporcionan un fondo y borde, mientras que las que tienen el sufijo "Icon" proporcionan un icono pequeño.

Además de los aspectos de icono, el componente Button también admite una propiedad `icon` estándar. La diferencia entre la propiedad estándar y la propiedad de estilo es que la propiedad de estilo permite establecer iconos para los estados individuales, mientras que la propiedad estándar sólo permite establecer un icono para todos los estados. Si una instancia de Button tiene establecidas la propiedad `icon` y propiedades de estilo de icono, es posible que no se comporte de la forma esperada.

Puede ver una demostración interactiva que ilustra cuándo se usa cada aspecto en el apartado Utilización de componentes de la Ayuda.

Utilización de ActionScript para dibujar aspectos del componente Button

Los aspectos predeterminados de los temas Halo y Sample usan el mismo elemento de aspecto para todos los estados y dibujan los gráficos mediante ActionScript. La implementación de Halo utiliza una extensión de la clase RectBorder y código personalizado de dibujo para dibujar los estados. En la implementación de Sample se utiliza el mismo aspecto y la misma clase de ActionScript que en el tema Halo, con distintas propiedades establecidas en ActionScript para alterar la apariencia del componente Button.

Para crear una clase de ActionScript con el fin de usarla como aspecto y proporcionar estados diferentes, el aspecto puede leer la propiedad de estilo `borderStyle` del aspecto y la propiedad `emphasized` del elemento principal para determinar el estado. En la tabla siguiente se muestra el estilo de borde que se establece para cada aspecto:

Propiedad	Estilo del borde
`falseUpSkin`	`falseup`
`falseDownSkin`	`falsedown`
`falseOverSkin`	`falserollover`
`falseDisabled`	`falsedisabled`
`trueUpSkin`	`trueup`
`trueDownSkin`	`truedown`
`trueOverSkin`	`truerollover`
`trueDisabledSkin`	`truedisabled`

Para crear un aspecto personalizado del componente Button de ActionScript:

1. Cree un nuevo archivo de clase de ActionScript.

   Para este ejemplo, asigne al archivo el nombre RedGreenBlueSkin.as.

2. Copie el siguiente código ActionScript al archivo:

```
import mx.skins.RectBorder;
import mx.core.ext.UIObjectExtensions;

class RedGreenBlueSkin extends RectBorder
{
  static var symbolName:String = "RedGreenBlueSkin";
  static var symbolOwner:Object = RedGreenBlueSkin;

  function size():Void
  {
    var c:Number; // color
    var borderStyle:String = getStyle("borderStyle");

    switch (borderStyle) {
      case "falseup":
      case "falserollover":
      case "falsedisabled":
        c = 0x7777FF;
        break;
      case "falsedown":
        c = 0x77FF77;
        break;
      case "trueup":
      case "truedown":
```

```
        case "truerollover":
        case "truedisabled":
          c = 0xFF7777;
          break;
      }

      clear();
      var thickness = _parent.emphasized ? 2 : 0;
      lineStyle(thickness, 0, 100);
      beginFill(c, 100);
      drawRect(0, 0, __width, __height);
      endFill();
    }

    // Requerido para los aspectos.
    static function classConstruct():Boolean
    {
      UIObjectExtensions.Extensions();
      _global.skinRegistry["ButtonSkin"] = true;
      return true;
    }
    static var classConstructed:Boolean = classConstruct();
    static var UIObjectExtensionsDependency = UIObjectExtensions;
}
```

Esta clase crea un cuadrado con el estilo de borde: un cuadro de color azul para los estados seleccionable, puntero del ratón encima y desactivado, un cuadro de color verde para el estado presionado normal y un cuadro de color rojo para el elemento secundario expandido. Dibuja un borde fino en el caso normal y un borde grueso si el botón está resaltado.

3. Guarde el archivo.
4. Cree un archivo FLA nuevo y guárdelo en la misma carpeta que el archivo AS.
5. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
6. Asígnele el nombre `ButtonSkin`.
7. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
8. Seleccione Exportar para ActionScript.

   El identificador se rellenará automáticamente con `ButtonSkin`.
9. Establezca la clase de AS 2.0 en `RedGreenBlueSkin`.
10. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
11. Arrastre un componente Button al escenario.
12. Seleccione Control > Probar película.

Utilización de clips de película para personalizar aspectos del componente Button

En el ejemplo anterior se ilustra la forma de usar una clase de ActionScript para personalizar el aspecto del componente Button, que es el método usado por los aspectos proporcionados en los temas Halo y Sample. Sin embargo, como el ejemplo utiliza cuadros coloreados sencillos, en este caso es más sencillo usar distintos símbolos de clip de película como aspectos.

Para crear símbolos de clip de película para aspectos de Button:

1. Cree un nuevo archivo FLA.
2. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
3. Asígnele el nombre `RedButtonSkin`.
4. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
5. Seleccione Exportar para ActionScript.

   El identificador se rellenará automáticamente con `RedButtonSkin`.
6. Establezca `mx.skins.SkinElement` como clase de AS 2.0.
7. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
8. Abra el símbolo nuevo para editarlo.
9. Utilice las herramientas de dibujo para crear un cuadro con relleno en rojo y línea negra.
10. Establezca un estilo de borde muy fino.
11. Defina el cuadro, incluido el borde, para que se posicione en (0,0) y tenga una anchura de 100 y una altura de 100.

    La clase `SkinElement` ajusta el tamaño del contenido si es necesario.
12. Repita los pasos 2 a 11, cree aspectos de color verde y azul, y asígneles los nombres correspondientes.
13. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
14. Arrastre un componente Button al escenario.
15. Establezca el valor de la propiedad `toggled` en `true` para ver los tres aspectos.
16. Copie el siguiente código ActionScript al panel Acciones con la instancia de Button seleccionada.

```
onClipEvent(initialize) {
  falseUpSkin = "BlueButtonSkin";
  falseDownSkin = "GreenButtonSkin";
  falseOverSkin = "BlueButtonSkin";
  falseDisabledSkin = "BlueButtonSkin";
  trueUpSkin = "RedButtonSkin";
  trueDownSkin = "RedButtonSkin";
  trueOverSkin = "RedButtonSkin";
  trueDisabledSkin = "RedButtonSkin";
}
```

17. Seleccione Control > Probar película.

Clase Button

Herencia MovieClip > Clase UIObject > Clase UIComponent > Clase SimpleButton > Button

Nombre de clase de ActionScript mx.controls.Button

Las propiedades de la clase Button permiten realizar lo siguiente en tiempo de ejecución: añadir un icono a un botón, crear una etiqueta de texto e indicar si el botón actúa como un botón de comando o como un conmutador.

Si una propiedad de la clase Button se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente Button utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.Button.version);
```

NOTA El código `trace(myButtonInstance.version);` devuelve `undefined`.

La clase del componente Button es distinta del objeto Button incorporado de ActionScript.

Resumen de métodos de la clase Button

No hay métodos exclusivos de la clase Button.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Button de la clase UIObject. Al llamar a estos métodos desde el objeto Button, debe utilizarse la forma *`buttonInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.

Método	Descripción
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Button de la clase UIComponent. Al llamar a estos métodos desde el objeto Button, debe utilizarse la forma *`buttonInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase Button

En la tabla siguiente se enumeran las propiedades de la clase Button.

Propiedad	Descripción
`Button.icon`	Especifica un icono para una instancia del botón.
`Button.label`	Especifica el texto que aparece en un botón.
`Button.labelPlacement`	Especifica la orientación del texto de la etiqueta con respecto a un icono.

Propiedades heredadas de la clase SimpleButton

En la tabla siguiente se enumeran las propiedades que hereda la clase Button de la clase SimpleButton. Al acceder a estas propiedades debe utilizarse la forma *buttonInstance.propertyName*.

Propiedad	Descripción
SimpleButton.emphasized	Indica si el botón tiene el aspecto de un botón de comando predeterminado.
SimpleButton.emphasizedStyleDeclaration	Declaración de estilos cuando la propiedad emphasized está definida en true.
SimpleButton.selected	Valor booleano que indica si el botón está seleccionado (true) o no (false). El valor predeterminado es false.
SimpleButton.toggle	Valor booleano que indica si el botón se comporta como un conmutador (true) o no (false). El valor predeterminado es false.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Button de la clase UIObject. Al acceder a estas propiedades desde el objeto Button, debe utilizarse la forma *buttonInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
UIObject.height	Sólo lectura; altura del objeto, expresada en píxeles.
UIObject.left	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.right	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Button de la clase UIComponent. Al acceder a estas propiedades desde el objeto Button, debe utilizarse la forma *`buttonInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase Button

No hay eventos exclusivos de la clase Button.

Eventos heredados de la clase SimpleButton

En la tabla siguiente se enumeran los eventos que hereda la clase Button de la clase SimpleButton.

Propiedad	Descripción
`SimpleButton.click`	Se difunde cuando se hace clic en un botón.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Button de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Button de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Button.icon

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

buttonInstance.icon

Descripción

Propiedad; cadena que especifica el identificador de vinculación de un símbolo de la biblioteca que se utilizará como icono en una instancia del botón. El icono puede ser un símbolo de clip de película o un símbolo gráfico con un punto de registro en la esquina superior izquierda. Si el icono es demasiado grande para encajarlo en el botón, deberá modificar el tamaño del botón; ni el botón ni el icono cambian de tamaño automáticamente. Si el icono es más grande que el botón, sobresaldrá por los bordes del botón.

Para crear un icono personalizado, cree un clip de película o un símbolo gráfico. Seleccione el símbolo en el escenario en el modo de edición de símbolos e introduzca 0 en los cuadros X e Y del inspector de propiedades. En el panel Biblioteca, seleccione el clip de película y elija Vinculación en el menú emergente Biblioteca. Seleccione Exportar para ActionScript e introduzca un identificador en el cuadro de texto Identificador.

El valor predeterminado es una cadena vacía (`""`) que indica que no hay icono.

Utilice la propiedad `labelPlacement` para definir la posición del icono con respecto al botón.

NOTA	El icono no aparece en el escenario de Flash. Debe elegir Control > Probar película para ver el icono.

Ejemplo

Con un botón en el escenario cuya instancia se denomina `my_button`, el siguiente código asigna como un icono a la instancia de Button el clip de película del panel Biblioteca que tiene el identificador de vinculación `happiness`:

```
my_button.icon = "happiness";
```

También puede crear el botón y asignar el icono completamente en ActionScript mediante el método `UIObject.createClassObject()` (debe haberse creado ya un icono para el botón con el identificador de vinculación `happiness`). En primer lugar, arrastre el componente Button desde el panel Componentes a la biblioteca del documento actual, de modo que el componente aparezca en la biblioteca pero no en el escenario. A continuación, en el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript:

```
this.createClassObject(mx.controls.Button, "my_button", 1, {icon:
  "happiness"});
```

Véase también

`Button.labelPlacement`

Button.label

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
buttonInstance.label
```

Descripción

Propiedad; especifica la etiqueta de texto para una instancia del botón. De forma predeterminada, la etiqueta aparece centrada en el botón. Cuando se llama a este método, se sustituye el parámetro de edición de etiquetas especificado en el inspector de propiedades o el inspector de componentes. El valor predeterminado es `"Button"`.

Ejemplo

Con un botón en el escenario con nombre de instancia `my_button`, el siguiente código define la etiqueta como "Test Button":

```
my_button.label = "Test Button";
```

También puede crear el botón y asignar la etiqueta completamente en ActionScript mediante el método `UIObject.createClassObject()`. En primer lugar, arrastre el componente Button desde el panel Componentes a la biblioteca del documento actual, de modo que el componente aparezca en la biblioteca pero no en el escenario. A continuación, en el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript:

```
this.createClassObject(mx.controls.Button, "my_button", 1, {label: "Test
  Button"});
```

Véase también

`Button.labelPlacement`

Button.labelPlacement

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

buttonInstance.labelPlacement

Descripción

Propiedad; define la posición de la etiqueta con respecto al icono. El valor predeterminado es `"right"`. Los siguientes son los cuatro valores posibles; el icono y la etiqueta están siempre centrados vertical y horizontalmente en el área de delimitación del botón:

- `"right"` La etiqueta aparece a la derecha del icono.
- `"left"` La etiqueta aparece a la izquierda del icono.
- `"bottom"` La etiqueta aparece debajo del icono.
- `"top"` La etiqueta aparece encima del icono.

Ejemplo

Con un botón en el escenario con nombre de instancia `my_button` y un símbolo en el panel Biblioteca con el identificador de vinculación `happiness`, el siguiente código define la alineación de la etiqueta a la izquierda del icono:

```
my_button.icon = "happiness";
my_button.label = "Test Button";
my_button.labelPlacement = "left";
```

También puede crear el botón y definir la alineación de la etiqueta completamente en ActionScript mediante el método `UIObject.createClassObject()`. En primer lugar, arrastre el componente Button desde el panel Componentes a la biblioteca del documento actual, de modo que el componente aparezca en la biblioteca pero no en el escenario. A continuación, en el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript:

```
this.createClassObject(mx.controls.Button, "my_button", 1, {label: "Test
  Button", icon: "happiness", labelPlacement: "left"});
```

CAPÍTULO 5

Interfaz API CellRenderer

5

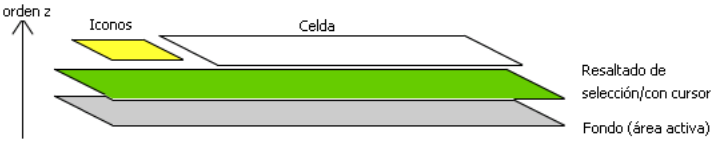
La interfaz API CellRenderer es un conjunto de propiedades y métodos que los componentes basados en listas (List, DataGrid, Tree, Menu y ComboBox) utilizan para manipular y mostrar el contenido de las celdas personalizadas en cada una de sus filas. Las celdas personalizadas pueden contener un componente creado previamente, como un componente CheckBox o cualquier clase que se cree.

Aspectos básicos de la clase List

Para utilizar la interfaz API CellRenderer, necesita conocimientos avanzados sobre la clase List. Los componentes DataGrid, Tree, Menu y ComboBox son extensiones de la clase List, así que cuando comprenda la clase List también comprenderá estas otras clases.

Información general sobre la composición del componente List

Los componentes List se componen de filas. Estas filas muestran los resaltados que aparecen al seleccionar o pasar el cursor sobre elementos, se utilizan como estados de acierto en la selección de filas y son un elemento fundamental en el desplazamiento. Además de los resaltados de selección y de los iconos (como los iconos de nodo y las flechas de ampliación de un componente Tree), una fila se compone de una celda (o, en el caso del componente DataGrid, de muchas celdas). De forma predeterminada, estas celdas son objetos TextField que implementan la API CellRenderer. No obstante, se puede indicar a un componente List que utilice una clase de componente diferente como celda para cada fila. La única condición es que la clase implemente la API CellRenderer, que utiliza el componente List para comunicarse con la celda.



Orden de apilamiento de una fila de un componente List o DataGrid

NOTA	Si una celda contiene controladores de evento de botón (`onPress`, etc.), puede que el área activa de fondo no reciba la información necesaria para activar los eventos.

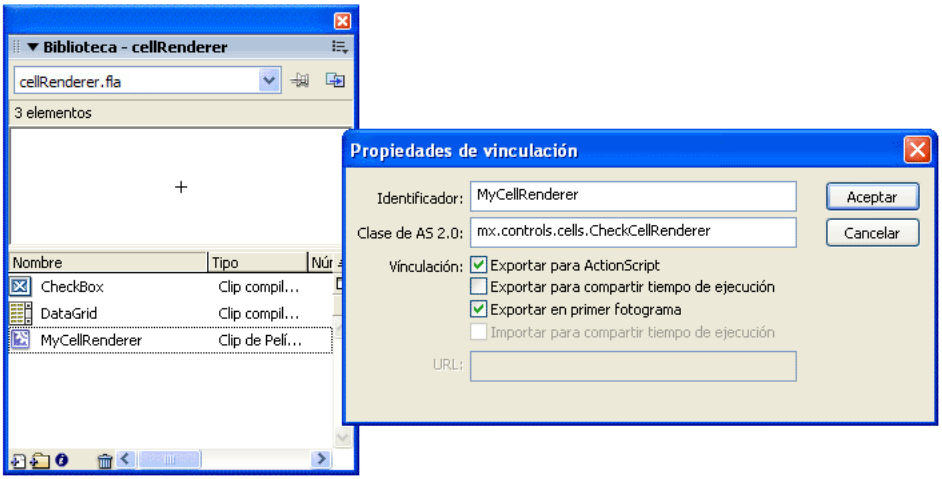
Comportamiento de desplazamiento del componente List

En la clase List se utiliza un algoritmo bastante complejo para el desplazamiento. Una lista sólo presenta las filas que puede mostrar de una vez; los elementos que excedan el valor de la propiedad `rowCount` no obtendrán filas. Al desplazarse por la lista, las filas se mueven hacia arriba o hacia abajo (en función del sentido del desplazamiento). Después, la lista recicla las filas que se salen de la vista, las reinicializa y las utiliza para las nuevas filas que se están desplazando por la vista. Para ello, establece el valor de la fila anterior al nuevo elemento en la vista y mueve la fila anterior al punto en que se desplaza el nuevo elemento en la vista.

Debido a este comportamiento, no se puede esperar que una celda se utilice para un único valor. El reciclaje de filas significa que el procesador de celdas debe saber cómo restablecer completamente su estado cuando se establece en un nuevo valor. Por ejemplo, si el procesador de celdas crea un icono para mostrar un elemento, deberá eliminar dicho icono cuando se procese otro elemento con él. Supongamos que el procesador de celdas es un contenedor que se va a ir rellenando con diferentes valores de elemento y que debe saber cómo pasar completamente de la visualización de un valor a la de otro. De hecho, es necesario que la celda sepa incluso cómo procesar correctamente elementos no definidos, lo cual puede significar eliminar contenido antiguo de la celda.

Utilización de la interfaz API CellRenderer

Debe escribir una clase con cuatro métodos (`CellRenderer.getPreferredHeight()`, `CellRenderer.getPreferredWidth()`, `CellRenderer.setSize()` y `CellRenderer.setValue()`) que el componente basado en una lista utiliza para comunicarse con la celda (si la clase amplía UIObject, puede utilizar `size()` en lugar de `CellRenderer.setSize()`). La clase debe especificarse en el cuadro de texto Clase de AS 2.0 en el cuadro diálogo Propiedades de vinculación de un símbolo de clip de película en la aplicación Flash.



Puede examinar la clase CheckCellRenderer que implementa la API del procesador de celdas para ver un ejemplo; se encuentra en First Run/classes/mx/controls/cells. Consulte también la documentación del componente DataGrid para obtener información relacionada con CellRenderer, como "Estrategias de rendimiento de DataGrid" en la página 269.

A la celda se le asignan automáticamente dos métodos y una propiedad (`CellRenderer.getCellIndex()`, `CellRenderer.getDataLabel()` y `CellRenderer.listOwner`) para que pueda comunicarse con el componente basado en lista. Por ejemplo, suponga que una celda contiene una casilla de verificación que, cuando se selecciona, hace que se seleccione una fila. El procesador de celdas necesita una referencia al componente basado en lista que lo contiene para llamar a la propiedad `selectedIndex` del componente. Además, la celda debe saber qué índice de elemento está representando actualmente para poder definir el número correcto en la propiedad `selectedIndex`; para ello, la celda puede utilizar `CellRenderer.listOwner` y `CellRenderer.getCellIndex()`. No es necesario implementar estos elementos de ActionScript; la celda los recibe automáticamente cuando se coloca en el componente basado en lista.

Ejemplo sencillo de procesador de celdas

En esta sección se presenta un ejemplo de procesador de celdas que muestra varias líneas de texto en una celda.

En el siguiente tutorial se muestra la forma de crear una clase de procesador de celdas que visualiza varias líneas de texto en las celdas de un componente DataGrid.

Los archivos completados, MultiLineCell.as y CellRenderer_tutorial.fla se encuentran en www.macromedia.com/go/component_samples_es.

Creación de la clase de procesador de celdas MultiLineCell

Una clase de procesador de celdas debe implementar los siguientes métodos:

- `CellRenderer.getPreferredHeight()`
- `CellRenderer.getPreferredWidth()`

  El método `CellRenderer.getPreferredWidth()` sólo es necesario para componentes Menu o encabezados de DataGrid; si no es el caso, márquelo como comentario, como se indica en el ejemplo.
- `CellRenderer.setSize()`

  Si una clase de procesador de celdas amplía UIObject, use `implement size()` en su lugar, como se indica en este ejemplo.
- `CellRenderer.setValue()`

Una clase de procesador de celdas también debe declarar los métodos y la propiedad recibidos de la clase List:

- `CellRenderer.getCellIndex()`
- `CellRenderer.getDataLabel()`
- `CellRenderer.listOwner`

Los pasos siguientes muestran la forma de crear un archivo de clase de procesador de celdas de ActionScript 2.0 denominado **MultiLineCell.as** y vincularlo a un nuevo símbolo de clip de película en un nuevo documento de Flash. A continuación, puede añadir un componente DataGrid a la biblioteca del documento de Flash. En el primer fotograma se añade código ActionScript para crear el objeto DataGrid dinámicamente y se asigna la clase MultiLineCell como procesador de celdas para una de sus columnas:

Para crear la clase de procesador de celdas multiLineCell:

1. En Flash, seleccione Archivo > Nuevo > Archivo ActionScript (no Documento de Flash). Guarde el documento como **MultiLineCell.as**.

2. Introduzca el código siguiente en MultiLineCell.as:

```
// ActionScript 2.0 class.
class MultiLineCell extends mx.core.UIComponent
{
  private var multiLineLabel; // La etiqueta que se va a utilizar para
  el texto.
  private var owner; // La fila que contiene esta celda.
  private var listOwner; // La lista, cuadrícula de datos o árbol que
  contiene esta celda.

  // Desplazamiento de la altura de la celda con respecto a la altura de
  la fila total y la anchura de celda preferida.
  private static var PREFERRED_HEIGHT_OFFSET = 4;
  private static var PREFERRED_WIDTH = 100;
  // Profundidad inicial.
  private var startDepth:Number = 1;

  // Constructor. Debe estar vacío.
  public function MultiLineCell()
  {
  }

  /* UIObject espera que llene createChildren creando instancias de
  todos los elementos de clip de película que podría necesitar tras la
  inicialización. En este caso vamos a crear una etiqueta*/
  public function createChildren():Void
  {
    // El método createLabel es un método útil de UIObject y una forma
  cómoda
    // de crear etiquetas en componentes.
    var c = multiLineLabel = this.createLabel("multiLineLabel",
  startDepth);
    // Vincula el estilo de la etiqueta al estilo de la cuadrícula
    c.styleName = listOwner;
    c.selectable = false;
    c.tabEnabled = false;
    c.background = false;
    c.border = false;
    c.multiline = true;
    c.wordWrap = true;
  }

  public function size():Void
  {
```

```
/* Si se amplía UIComponent, que importa UIObject, se obtiene setSize
  automáticamente; no obstante, UIComponent espera que se implemente
  size(). Suponga que __width y __height se establecen ahora. Va a
  ampliar la celda para adaptarla a rowHeight. rowHeight es una
  propiedad del componente de tipo lista en el que estamos procesando
  una celda. Como queremos ajustar el valor de rowHeight en dos líneas,
  al crear el componente de tipo lista con esta clase cellRenderer,
  asegúrese de que el valor de su propiedad rowHeight es suficientemente
  grande para que dos líneas de texto puedan representarse en ella.*/

/*__width y __height son las variables subyacentes de los captadores/
  definidores .width y .height.*/
    var c = multiLineLabel;
    c.setSize(__width, __height);
  }

  // Proporciona la altura preferida de la celda. Método heredado.
  public function getPreferredHeight():Number
  {
/* Se asigna a la celda una propiedad, "owner", que hace referencia a la
  fila. Es siempre preferible que la celda ocupe la mayor parte de la
  altura de la fila. En este caso mantendremos la celda ligeramente más
  pequeña.*/
    return owner.__height - PREFERRED_HEIGHT_OFFSET;
  }

  // Lo llama el propietario para establecer el valor en la celda.
  Método heredado.
  public function setValue(suggestedValue:String, item:Object,
  selected:Boolean):Void
  {
/* Si el elemento no está definido, no debe procesarse nada en la celda,
  y esta etiqueta debe establecerse como invisible. Nota: para el
  desplazamiento de componentes de tipo List, como una cuadrícula de
  datos con desplazamiento, las celdas se vacían cuando dejan de estar
  visibles y a continuación se reutilizan y se les asigna un nuevo
  valor, lo que produce un efecto animado de desplazamiento. Por esta
  razón, no puede dar por supuesto que una celda siempre tendrá datos o
  mostrará siempre el mismo valor.*/
    if (item!=undefined){
      multiLineLabel.text._visible = false;
    }
    multiLineLabel.text = suggestedValue;
  }
  // función getPreferredWidth :: sólo para menús y encabezados de
  cuadrícula de datos
  // función getCellIndex :: no se usa en este procesador de celdas
  // función getDataLabel :: no se usa en este procesador de celdas
}
```

Creación de una aplicación para probar la clase de procesador de celdas MultiLineCell

En los pasos siguientes creará la instancia de DataGrid e implementará la clase MultiLineCell.

Para crear una aplicación con un componente DataGrid que utiliza la clase de procesador de celdas MultiLineCell:

1. En Flash, seleccione Archivo > Nuevo > Documento de Flash.
2. Seleccione Archivo > Guardar como, asigne al archivo el nombre **cellRender_tutorial.fla** y guárdelo en la misma carpeta que el archivo MultiLineCell.as.
3. Para crear un nuevo símbolo de movieClip a fin de vincularlo a la clase MultiLineCell, seleccione Insertar > Nuevo símbolo.
4. Haga clic en el botón Avanzado en la esquina inferior derecha del cuadro de diálogo Crear un nuevo símbolo para activar más opciones.

   El botón Avanzado está disponible en el modo básico del cuadro de diálogo Crear un nuevo símbolo. Si no ve el botón Avanzado, probablemente ya esté en la vista avanzada del cuadro de diálogo.
5. En el cuadro de texto Nombre, escriba **MultiLineCell**.

   El valor predeterminado de Tipo es Clip de película. Deje activada la opción Clip de película.
6. Active la casilla de verificación Exportar para ActionScript en la sección Vinculación.

   La activación de esta opción le permite asociar dinámicamente instancias de este símbolo a los documentos de Flash durante la ejecución. El cuadro de texto Identificador mostrará automáticamente MultiLineCell.
7. Establezca la Clase de ActionScript 2.0 en MultiLineCell (para que coincida con el nombre de la clase del procesador de celdas MultiLineCell creada previamente).
8. Active la casilla de verificación Exportar en primer fotograma y haga clic en Aceptar para aplicar los cambios y cerrar el cuadro de diálogo.

NOTA	Si necesita modificar posteriormente las propiedades de vinculación del símbolo Clip de película de MultiLineCell, puede hacer clic con el botón derecho en el símbolo en la biblioteca del documento y seleccionar Propiedades o Vinculación desde el menú.
9. Arrastre un componente DataGrid desde el panel Componentes a la biblioteca.

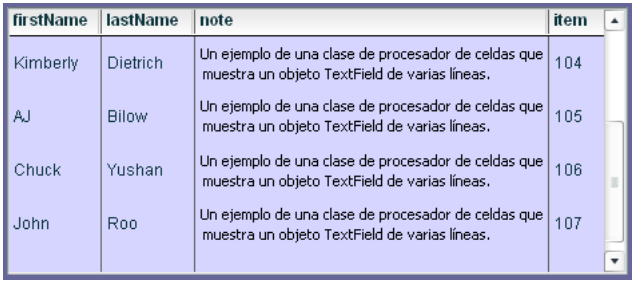
   La instancia de DataGrid se creará dinámicamente mediante ActionScript en el siguiente paso.
10. Seleccione el primer fotograma de la línea de tiempo principal (asegúrese de que no sigue en el modo de edición de clip de película de MultiLineCell).

11. En el panel Acciones para el primer fotograma, introduzca el código siguiente para crear dinámicamente una instancia de DataGrid, asigne datos a la instancia de DataGrid y asigne la nueva clase de procesador de celdas:

```
// Crear una nueva instancia del componente DataGrid
this.createClassObject(mx.controls.DataGrid, "myGrid_dg", 1);

// Generar un proveedor de datos para la cuadrícula de datos con cuatro
  columnas de datos.
myDP = new Array();
var aLongString:String = "An example of a cell renderer class that
  displays a multiple line TextField";
myDP.addItem({firstName:"Winston", lastName:"Elstad", note:aLongString,
  item:100});
myDP.addItem({firstName:"Ric", lastName:"Dietrich", note:aLongString,
  item:101});
myDP.addItem({firstName:"Ewing", lastName:"Canepa", note:aLongString,
  item:102});
myDP.addItem({firstName:"Kevin", lastName:"Wade", note:aLongString,
  item:103});
myDP.addItem({firstName:"Kimberly", lastName:"Dietrich",
  note:aLongString, item:104});
myDP.addItem({firstName:"AJ", lastName:"Bilow", note:aLongString,
  item:105});
myDP.addItem({firstName:"Chuck", lastName:"Yushan", note:aLongString,
  item:106});
myDP.addItem({firstName:"John", lastName:"Roo", note:aLongString,
  item:107});

/* Asignar el proveedor de datos al objeto DataGrid para llenarlo. Nota:
  hay que hacer esto antes de aplicar los procesadores de celdas. */
myGrid_dg.dataProvider = myDP;

/* Establecer algunas propiedades básicas de la cuadrícula. Nota: la
  altura de las filas de la cuadrícula de datos debe reflejar el número
  de líneas que se espera mostrar en el procesador de celdas
  MultiLineCell. El procesador de celdas ajustará el tamaño a la altura
  de la fila. Debe ser alrededor de 40 para 2 líneas o de 60 para 3
  líneas con el tamaño de texto predeterminado.*/
myGrid_dg.setSize(430,200);
myGrid_dg.move(40,40);
myGrid_dg.rowHeight = 40; // Permite 2 líneas de texto con el tamaño de
  texto predeterminado.
myGrid_dg.getColumnAt(0).width = 70;
myGrid_dg.getColumnAt(1).width = 70;
myGrid_dg.getColumnAt(2).width = 220;
myGrid_dg.resizableColumns = true;
myGrid_dg.vScrollPolicy = "auto";
myGrid_dg.setStyle("backgroundColor", 0xD5D5FF);

// Asignar procesadores de celdas.
myGrid_dg.getColumnAt(2).cellRenderer = "MultiLineCell";
```

12. Guarde el documento de Flash y seleccione Control > Probar película.

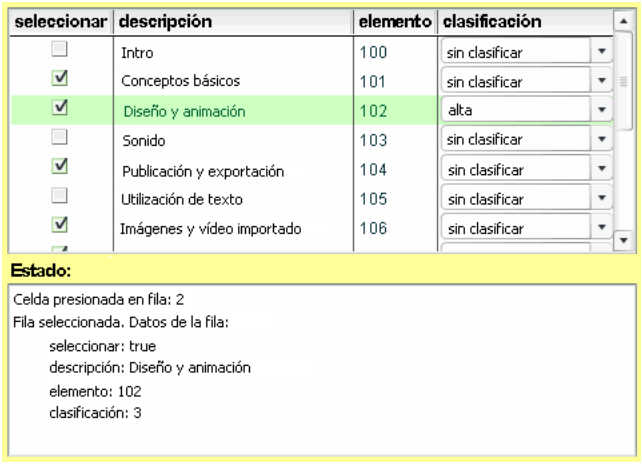
Aparece una cuadrícula de datos. La tercera columna de la cuadrícula de datos contiene una celda de varias líneas.



El ejemplo de procesador de celdas MultiLineCell completado.

Ejemplos adicionales de procesador de celdas

También se proporcionan ejemplos adicionales de clases de procesador de celdas que visualizan componentes ComboBox y CheckCell. Estos archivos se encuentran en la carpeta CellRenderers_sample, dentro de la carpeta Samples and Tutorials del disco duro, en www.macromedia.com/go/component_samples_es.



El ejemplo instalado adicional, denominado CellRenderers_Sample, que muestra controles ComboBox y CheckBox.

Métodos que se deben implementar para la API CellRenderer

Debe escribir una clase con los métodos siguientes para que el componente List, DataGrid, Tree o Menu pueda comunicarse con la celda.

Método	Descripción
`CellRenderer.getPreferredHeight()`	Devuelve la altura preferida de una celda.
`CellRenderer.getPreferredWidth()`	La anchura preferida de una celda.
`CellRenderer.setSize()`	Define la anchura y la altura de una celda.
`CellRenderer.setValue()`	Define el contenido que va a mostrarse en la celda.

Métodos proporcionados por la interfaz API CellRenderer

Los componentes List, DataGrid, Tree y Menu proporcionan los siguientes métodos a la celda cuando se crea en el componente. No es necesario implementar estos métodos.

Método	Descripción
`CellRenderer.getCellIndex()`	Devuelve un objeto con dos campos, `columnIndex` y `itemIndex`, que indican la posición de la celda.
`CellRenderer.getDataLabel()`	Devuelve una cadena que contiene el nombre del campo de datos del procesador de celdas.

Propiedades proporcionadas por la interfaz API CellRenderer

Los componentes List, DataGrid, Tree y Menu proporcionan las siguientes propiedades a la celda cuando se crea en el componente. No es necesario implementar estas propiedades.

Propiedad	Descripción
`CellRenderer.listOwner`	Referencia al componente List que contiene la celda.
`CellRenderer.owner`	Referencia a la fila que contiene la celda.

CellRenderer.getCellIndex()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.getCellIndex()

Parámetros

Ninguno.

Valor devuelto

Un objeto con dos campos: `columnIndex` e `itemIndex`.

Descripción

Método; devuelve un objeto con dos campos, `columnIndex` e `itemIndex`, que ubican la celda en el componente. Cada campo es un entero que indica la posición de columna y la posición de elemento de una celda. El valor de `columnIndex` es siempre 0, excepto para el componente DataGrid.

Este método lo proporciona la clase List; no es necesario implementarlo. Puede declararlo en la clase de procesador de celdas de la manera que se indica a continuación y usarlo en las funciones del procesador de celdas:

```
var getCellIndex:Function;
```

Ejemplo

En este ejemplo se edita el proveedor de datos de un componente DataGrid desde una celda:

```
var index = getCellIndex();
var colName = listOwner.getColumnAt(index.columnIndex).columnName;
listOwner.dataProvider.editField(index.itemIndex, colName, someVal);
```

CellRenderer.getDataLabel()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.getDataLabel()
```

Parámetros

Ninguno.

Valor devuelto

Una cadena.

Descripción

Método; devuelve una cadena que contiene el nombre del campo de datos del procesador de celdas. Para el componente DataGrid, este método devuelve el nombre de columna para la celda actual.

Este método lo proporciona la clase List; no es necesario implementarlo. Puede declararlo en la clase de procesador de celdas de la manera que se indica a continuación y usarlo en las funciones del procesador de celdas:

```
var getDataLabel:Function;
```

Ejemplo

El código siguiente indica a la celda el nombre del campo de datos que está procesando. Por ejemplo, si el nombre del campo de datos que la celda está procesando actualmente es `"Price"`, el valor actual de la variable `p` es `"Price"`:

```
var p = getDataLabel();
```

CellRenderer.getPreferredHeight()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.getPreferredHeight()

Parámetros

Ninguno.

Valor devuelto

La altura correcta de la celda.

Descripción

Método; devuelve la altura preferida de una celda. Es especialmente importante para obtener la altura correcta del texto de la celda. Si define un valor más alto que el de la propiedad `rowHeight` del componente, la celda se extenderá sobre las filas hacia arriba y hacia abajo.

Este método no lo proporciona la clase List; debe implementarlo. Indica a las filas de la lista cómo centrar la celda y cómo ajustar la altura de las celdas si es necesario. Asimismo, si es necesario, puede devolver una constante (por ejemplo, 22) o medir y devolver la altura del contenido. También puede devolver `owner.height`, que es la altura de la fila.

Ejemplo

En este ejemplo se devuelve el valor 20, lo cual indica que la celda debe tener una altura de 20 píxeles.

```
function getPreferredHeight(Void) :Number
{
   return 20;
}
```

En este ejemplo se devuelve un valor inferior en 4 píxeles a la altura de la fila:

```
function getPreferredHeight():Number
{
/* Sabe que se asigna a la celda una propiedad, "owner", que es la fila. Es
  siempre preferible que la celda ocupe la mayor parte de la altura de la
  fila.
*/
return owner.__height - 4;
}
```

CellRenderer.getPreferredWidth()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.getPreferredWidth()
```

Parámetros

Ninguno.

Valor devuelto

Un valor (de tipo Number) que indica la anchura correcta de la celda.

Descripción

Método; devuelve la anchura preferida de una celda. Si especifica una anchura mayor que la del componente, la celda podría verse cortada.

Implemente este método para el componente Menu. La celda se ajustará a la anchura de la fila, salvo en un menú, que debe medir el texto para la anchura de la fila. También puede implementar este método para el componente DataGrid en el que el procesador de encabezados comprueba si se debe mostrar o no la flecha de ordenación.

Ejemplo

Este ejemplo devuelve el valor multiplicado por 3, que indica que la celda debe ser tres veces mayor que la longitud de la cadena que representa:

```
function getPreferredWidth():Number
{
   return myString.length*3;
}
```

En este ejemplo se marca como comentario el método `getPreferredWidth()`:

```
// function getPreferredWidth :: sólo para un menú o una cuadrícula de datos
```

CellRenderer.listOwner

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.listOwner

Descripción

Propiedad; referencia a la lista propietaria de la celda. Esa lista puede ser un componente DataGrid, Tree, List o Menu.

Este método lo proporciona la clase List; no es necesario implementarlo. Declárelo en la clase del procesador de celdas como se indica a continuación y úselo como una referencia a la lista (o el árbol, menú o cuadrícula):

```
var listOwner:MovieClip; // o UIObject, etc.
```

Ejemplo

En este ejemplo se busca el elemento seleccionado de la lista en una celda:

```
var s = listOwner.selectedItem;
```

CellRenderer.owner

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.owner

Descripción

Propiedad; referencia a la fila que contiene la celda.

Este método lo proporciona la clase List; no es necesario implementarlo. Declárelo en la clase del procesador de celdas y úselo como una referencia:

```
var owner:MovieClip; // o UIObject, etc.
```

CellRenderer.setSize()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.setSize(*width*, *height*)

Parámetros

width Número que indica la anchura con la que se diseña el componente.

height Número que indica la altura con la que se diseña el componente.

Valor devuelto

Ninguno.

Descripción

Método; permite que la lista indique a sus celdas el tamaño con el que deben mostrarse. El procesador de celdas debe realizar el diseño para que quepa en el área especificada a fin de que las celdas no se extiendan a otras partes de la lista y aparezcan rotas.

Si el procesador de celdas amplía la clase UIObject, debe implementar en su lugar el método `size()`. Escriba la misma función que escribiría para `setSize()`, pero use las propiedades `width` y `height` en lugar de parámetros.

Ejemplo

En el siguiente ejemplo se redimensiona una imagen que aparece en la celda para que quepa en los límites especificados por la lista:

```
function setSize(w:Number, h:Number):Void
{
   image._width = w-2;
   image._height = h-2;
   image._x = image._y = 1;
}
```

Este ejemplo está en una clase de procesador de celdas que amplía UIComponent (que amplía UIObject), por lo que debe implementar `size()` en lugar de `setSize()` de la manera siguiente:

```
// Al ampliar UIComponent, obtiene setSize;
// sin embargo, UIComponent espera que implemente size().
// Suponga que __width y __height se establecen ahora.
// Va a ampliar la celda para adaptarla a rowHeight.

function size():Void
{
// __width y __height son las variables subyacentes
// de las propiedades .width y .height de los captadores/definidores
  var c = multiLineLabel;
  c._width = __width;
  c._height = __height;
}
```

CellRenderer.setValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.setValue(*suggested*, *item*, *selected*)

Parámetros

suggested Valor que debe utilizarse para el texto del procesador de celdas, si procede.

item Objeto que es el elemento completo que debe procesarse. El procesador de celdas puede usar las propiedades de este objeto para el procesamiento.

selected Cadena con los siguientes valores posibles: `"normal"`, `"highlighted"` y `"selected"`.

Valor devuelto

Ninguno.

Descripción

Método; toma los valores dados y crea una representación de ellos en la celda. Esto borra las diferencias entre lo que se mostraba en la celda y lo que se debe mostrar para el nuevo elemento. Recuerde que cualquier celda puede mostrar muchos valores mientras esté visible en la lista. Éste es el método más importante de CellRenderer y debe implementarse en cada procesador de celdas.

El método `setValue()` se usa con frecuencia (por ejemplo, al desplazar el puntero del ratón sobre un elemento, en una selección, al cambiar el tamaño de una columna o cuando se produce un desplazamiento). Es importante recordar que una celda podría no existir en el escenario y que no siempre debe actualizarse con datos cuando se llama a `setValue()`. Por ejemplo, una celda concreta puede salir en cualquier momento del área de visualización o reutilizarse para procesar otro valor. Por esta razón, no puede hacer referencia directamente a una instancia específica de procesador de celdas en la cuadrícula y debe escribir sentencias `if` en el cuerpo de `setValue()` que permitan al código ejecutarse únicamente si el parámetro `item` está definido y se ha producido un cambio. Un parámetro item sin definir indica que la celda debe estar visible y vacía, y que se debe asignar a la propiedad `_visible` de los elementos de la celda el valor `false`. Se podría requerir que una celda estuviera visible y vacía temporalmente, como cuando se realiza un desplazamiento en un componente DataGrid.

Si se selecciona una fila y el puntero del ratón está encima de la fila, el valor del parámetro seleccionado será `"highlighted"`, no `"selected"`. Esto puede provocar problemas si se intenta hacer que el procesador de celdas se comporte de manera diferente en función de si la fila está en un estado seleccionado. Para comprobar si la fila actual está en un estado seleccionado, use el código siguiente:

```
var reallySelected:Boolean = selected != "normal" && listOwner.selectedNode
  == item;
```

Ejemplo

En el ejemplo siguiente se muestra la forma de usar `setValue()` y `editField()` para hacer referencia a una instancia de procesador de celdas en una cuadrícula.

Como una celda podría no existir en el escenario (puede desplazarse fuera del área de visualización o podría reutilizarse para procesar otro valor) en cualquier momento, no puede hacer referencia directamente a una instancia de procesador de celdas específico en la cuadrícula.

En su lugar, use el proveedor de datos para comunicarse con una celda específica de la cuadrícula. El proveedor de datos contiene toda la información de estado sobre la cuadrícula. Para mostrar una celda determinada como activada o seleccionada (marcada), debe haber un campo correspondiente en el proveedor de datos que contenga esa información. El método `setValue()` del procesador de celdas comunica los cambios de estado del proveedor de datos a la celda. A continuación se muestra una implementación del método `setValue()` de un procesador de celdas teórico que procesa una casilla de verificación en las celdas:

```
function setValue(str, itm, sel)
{
/* Suponga que el proveedor de datos tiene dos campos relevantes para esta
  celda : checked y enabled.
La forma de este proveedor de datos podría ser:
[
{field1:"DisplayMe", field2:"SomeString", checked:true, enabled:false}
{field1:"DisplayMe", field2:"SomeString", checked:false, enabled:true}
{field1:"DisplayMe", field2:"SomeString", checked:true, enabled:true}
]
*/

/* Ocultar algo que normalmente se procesa en la celda si el valor de item
  es undefined. Si no, actualizar el contenido de la celda con los nuevos
  datos.
*/
  if (itm == undefined){
    myCheck._visible = false;
  }else{

    // comprobación de redundancia
    if (myCheck.selected!=itm.checked){
      myCheck.selected = itm.checked;
    }
    if (myCheck.enabled!=itm.enabled){
      myCheck.enabled = itm.enabled;
    }
  }
}
```

Si desea activar la casilla de verificación en la segunda fila, debe comunicarse a través del proveedor de datos. Cualquier cambio que se produzca en el proveedor de datos (si se realiza mediante un método DataProvider como `DataProvider.editField()`) llama a `setValue()` para actualizar la presentación de la cuadrícula. Este código se escribiría en la aplicación Flash, en un fotograma, un objeto u otro archivo de clase (pero no en el archivo de clase del procesador de celdas):

```
// vuelve a llamar a setValue()
myGrid.editField(1, "enabled", true);
```

En el ejemplo siguiente se carga una imagen en un componente Loader dentro de la celda, según el valor que se haya pasado:

```
function setValue(suggested, item, selected) : Void
{
/* Ocultar algo que normalmente se procesa en la celda si el valor de item
  es undefined. Si no, actualizar el contenido de la celda con los nuevos
  datos.
*/
  if (item == undefined){
    loader._visible = false;
  }else{
    // borrar el componente Loader
    loader.contentPath = undefined;
    // la lista tiene URL de distintas imágenes en su proveedor de datos
    if (suggested!=undefined){
      loader.contentPath = suggested;
    }
  }
}
```

El ejemplo siguiente es de un procesador de celdas de texto de varias líneas:

```
function setValue(suggested:String, item:Object, selected:Boolean):Void
{
/* Ocultar algo que normalmente se procesa en la celda si el valor de item
  es undefined. Si no, actualizar el contenido de la celda con los nuevos
  datos.
*/
  if (item == undefined){
    multiLineLabel._visible = false;
  }else{
    // añade el texto a la etiqueta
    multiLineLabel.text = suggested;
  }
}
```

El ejemplo siguiente es de un procesador de botones de opción. Si el valor del parámetro item es undefined, se puede desplazar la celda fuera del área de visualización y debe estar visible y vacía. Se utiliza una sentencia `if` para determinar si el valor del parámetro `item` es undefined. Si el valor del parámetro `item` es undefined, el botón de opción se oculta estableciendo el valor de su propiedad `_visible` en `false`; de lo contrario, el botón de opción se actualiza con los nuevos datos y vuelve a estar visible.

```
function setValue(str:String, item:Object, sel:String) : Void {
/* Ocultar algo que normalmente se procesa en la celda si el valor de item
  es undefined. Si no, actualizar el contenido de la celda con los nuevos
  datos.
*/
  if (item == undefined) {
    radio._visible = false; }
  else {
    trace(item.data + " " + item.label + " " + item.state + " " + sel);
    radio.label = item.label;
    radio.data = item.data;
    radio.selected = item.state;
    radio._visible = true;
  }
}
```

CAPÍTULO 6

Componente CheckBox

6

Una casilla de verificación es un cuadrado que se puede seleccionar y cuya selección se puede anular. Cuando se selecciona, en el cuadro aparece una marca. A una casilla de verificación se le puede añadir una etiqueta de texto, que puede colocarse a la izquierda, derecha, arriba o abajo.

Una casilla de verificación puede estar activada o desactivada en una aplicación. Si está activada y el usuario hace clic en ella o en su etiqueta, la casilla recibe la selección de entrada y muestra su aspecto presionado. Si el usuario desplaza el puntero fuera del área de delimitación de la casilla o de su etiqueta mientras presiona el botón del ratón, el componente recupera su aspecto original y conserva la selección de entrada. El estado de una casilla de verificación no cambia hasta que se suelta el ratón sobre el componente. Además, una casilla de verificación tiene dos estados desactivados (seleccionado y no seleccionado) que no permiten la interacción con el ratón ni el teclado.

Si la casilla de verificación está desactivada, muestra su aspecto desactivado sea cual sea la acción del usuario. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado.

La instancia de CheckBox recibe la selección cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando una instancia de CheckBox está seleccionada, se pueden utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Mayús+Tabulador	Desplaza la selección al elemento anterior.
Barra espaciadora	Selecciona o anula la selección del componente y activa el evento `click`.
Tabulador	Desplaza la selección al elemento siguiente.

Para más información sobre el control de la selección, consulte “Clase FocusManager” en la página 745 o “Creación de un desplazamiento personalizado de la selección” en *Utilización de componentes*.

La previsualización dinámica de cada instancia de CheckBox refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes.

Cuando se añade el componente CheckBox a una aplicación, se puede utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.CheckBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Utilización del componente CheckBox

Una casilla de verificación es una parte fundamental de cualquier formulario o aplicación Web. Utilice casillas de verificación siempre que necesite reunir un conjunto de valores `true` o `false` que no se excluyan mutuamente. Por ejemplo, un formulario que recopila información personal sobre un cliente podría contener una lista de aficiones que el cliente debe elegir; cada afición llevaría al lado una casilla de verificación.

Parámetros de CheckBox

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente CheckBox en el inspector de propiedades o en el inspector de componentes:

label define el valor del texto de la casilla de verificación; el valor predeterminado es `CheckBox`.

labelPlacement orienta el texto de la etiqueta en la casilla de verificación. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom`; el valor predeterminado es `right`. Para más información, consulte `CheckBox.labelPlacement`.

selected define el valor inicial de la casilla de verificación en marcada (`true`) o sin marcar (`false`). El valor predeterminado es `false`.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales del componente CheckBox utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase CheckBox" en la página 142.

Creación de aplicaciones con el componente CheckBox

El siguiente procedimiento explica cómo añadir un componente CheckBox a una aplicación durante la edición. El ejemplo siguiente es un formulario para una aplicación de citas en línea. El formulario es un cuestionario que busca citas posibles para el cliente. El formulario de consulta debe tener una casilla de verificación con la etiqueta Restrict Age que permita a los clientes restringir la búsqueda a un grupo de edades especificado. Cuando se selecciona la casilla de verificación Restrict Age, el cliente puede introducir las edades mínima y máxima en dos campos de texto. (Estos campos de texto sólo se activan cuando la casilla de verificación está seleccionada.)

Para crear una aplicación con el componente CheckBox:

1. Arrastre dos componentes TextInput desde el panel Componentes al escenario.
2. En el inspector de propiedades, escriba los nombres de instancia **minimumAge** y **maximumAge**.
3. Arrastre un componente CheckBox desde el panel Componentes al escenario.
4. En el inspector de propiedades, siga este procedimiento:
   - Introduzca **restrictAge** como nombre de instancia.
   - Introduzca **Restrict Age** como parámetro labels.
5. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
var restrictAgeListener:Object = new Object();
restrictAgeListener.click = function (evt:Object) {
  minimumAge.enabled = evt.target.selected;
  maximumAge.enabled = evt.target.selected;
};
restrictAge.addEventListener("click", restrictAgeListener);
```

   Este código crea un controlador de eventos `click` que activa y desactiva los componentes de los campos de texto `minimumAge` y `maximumAge`, ya situados en el escenario. Para más información, consulte `CheckBox.click` y "Componente TextInput" en la página 1247.

Para crear una casilla de verificación mediante código ActionScript:

1. Arrastre el componente CheckBox desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. Arrastre el componente TextInput desde el panel Componentes a la biblioteca del documento actual.

3. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones para crear instancias de componente y colocarlas:

```
this.createClassObject(mx.controls.CheckBox, "testAge_ch", 1,
  {label:'Age Range', selected:true});
this.createClassObject(mx.controls.TextInput, "minimumAge_ti", 2,
  {restrict:[0-9], text:18, maxChars:2});
minimumAge_ti.move(20, 30);
this.createClassObject(mx.controls.TextInput, "maximumAge_ti", 3,
  {restrict:[0-9], text:55, maxChars:2});
maximumAge_ti.move(20, 60);
```

   Este script utiliza el método "UIObject.createClassObject()" en la página 1404 para crear la instancia de CheckBox denominada **restrictAge** y especifica una propiedad de etiqueta. A continuación, el código utiliza el método "UIObject.move()" en la página 1417 para colocar el botón.

4. Añada el siguiente código ActionScript para crear un detector de eventos y una función de controlador de eventos:

```
// Crear controlador para el evento checkBox.
function checkboxHandler(evt_obj:Object) {
 minimumAge_ti.enabled = evt_obj.target.selected;
 maximumAge_ti.enabled = evt_obj.target.selected;
}
// Añadir detector.
testAge_ch.addEventListener("click", checkboxHandler);
```

   Este código crea un controlador de eventos `click` que activa y desactiva los componentes de los campos de texto `minimumAge` y `maximumAge`. Para más información, consulte `CheckBox.click`, "EventDispatcher.addEventListener()" en la página 516 y "Componente TextInput" en la página 1247.

Personalización del componente CheckBox

El componente CheckBox puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o cualquier método o propiedad aplicable de la Clase CheckBox. Cuando se modifica el tamaño de la casilla de verificación, no cambia el tamaño de la etiqueta ni del icono de la casilla; sólo cambia el tamaño del recuadro de delimitación.

El recuadro de delimitación de una instancia de CheckBox es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

Utilización de estilos con el componente CheckBox

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de CheckBox. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

Un componente CheckBox admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).

Estilo	Tema	Descripción
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`symbolBackgroundColor`	Sample	Color de fondo de la casilla de verificación. El valor predeterminado es 0xFFFFFF (blanco).
`symbolBackgroundDisabledColor`	Sample	Color de fondo de la casilla de verificación cuando está desactivada. El valor predeterminado es 0xEFEEEF (gris claro).
`symbolBackgroundPressedColor`	Sample	Color de fondo de la casilla de verificación cuando se presiona. El valor predeterminado es 0xFFFFFF (blanco).
`symbolColor`	Sample	Color de la marca de verificación. El valor predeterminado es 0x000000 (negro).
`symbolDisabledColor`	Sample	Color de la marca de verificación desactivada. El valor predeterminado es 0x848384 (gris oscuro).

Utilización de aspectos con el componente CheckBox

El componente CheckBox utiliza símbolos de la biblioteca para representar estados de botón. Para aplicar aspectos al componente CheckBox durante la edición, modifique los símbolos del panel Biblioteca. Los aspectos del componente CheckBox se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/CheckBox Assets/states de la biblioteca del archivo HaloTheme.fla o SampleTheme.fla. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

El componente CheckBox utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
`falseUpSkin`	Estado sin presionar (normal) sin marcar. El valor predeterminado es `CheckFalseUp`.
`falseDownSkin`	Estado presionado sin marcar. El valor predeterminado es `CheckFalseDown`.
`falseOverSkin`	Estado sin marcar cuando se desplaza el puntero sobre él. El valor predeterminado es `CheckFalseOver`.
`falseDisabledSkin`	Estado desactivado sin marcar. El valor predeterminado es `CheckFalseDisabled`.
`trueUpSkin`	Estado conmutado marcado. El valor predeterminado es `CheckTrueUp`.
`trueDownSkin`	Estado presionado marcado. El valor predeterminado es `CheckTrueDown`.
`trueOverSkin`	Estado marcado cuando se desplaza el puntero sobre él. El valor predeterminado es `CheckTrueOver`.
`trueDisabledSkin`	Estado desactivado marcado. El valor predeterminado es `CheckTrueDisabled`.

Cada uno de estos aspectos se corresponde con el icono que indica el estado del componente CheckBox. El componente CheckBox no tiene bordes ni fondo.

Para crear símbolos de clip de película para aspectos de CheckBox:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.

3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta CheckBox Assets a la biblioteca del documento.
4. Expanda la carpeta CheckBox Assets/States en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo CheckFalseDisabled.
6. Personalice el símbolo como desee.

   Por ejemplo, cambie el color blanco del cuadrado interior por un gris claro.
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.

   Por ejemplo, repita el cambio de color en el cuadro interior del símbolo CheckTrueDisabled.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente CheckButton al escenario.

   En este ejemplo, arrastre dos instancias para mostrar los dos nuevos símbolos de aspecto.
10. Defina las propiedades de la instancia de CheckButton como desee.

    Para este ejemplo, establezca una instancia de CheckBox en `true` y use ActionScript para establecer ambas instancias de CheckBox en disabled.
11. Seleccione Control > Probar película.

Clase CheckBox

Herencia MovieClip > Clase UIObject > Clase UIComponent > Clase SimpleButton > Componente Button > CheckBox

Nombre de clase de ActionScript mx.controls.CheckBox

Las propiedades de la clase CheckBox permiten crear etiquetas de texto y colocarlas a la izquierda, derecha, encima o debajo de una casilla de verificación en tiempo de ejecución.

Si una propiedad de la clase CheckBox se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente CheckBox utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte “Creación de un desplazamiento personalizado de la selección” en *Utilización de componentes*.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.CheckBox.version);
```

NOTA	El código `trace(myCheckBoxInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase CheckBox

No hay métodos exclusivos de la clase CheckBox.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase CheckBox de la clase UIObject. Al llamar a estos métodos desde el objeto CheckBox, debe utilizarse la forma *`checkBoxInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase CheckBox de la clase UIComponent. Al llamar a estos métodos desde el objeto CheckBox, debe utilizarse la forma *checkBoxInstance.methodName*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase CheckBox

En la tabla siguiente se enumeran las propiedades de la clase CheckBox.

Propiedad	Descripción
`CheckBox.label`	Especifica el texto que aparece junto a una casilla de verificación.
`CheckBox.labelPlacement`	Especifica la orientación del texto de la etiqueta con respecto a la casilla de verificación.
`CheckBox.selected`	Especifica si la casilla de verificación está seleccionada (`true`) o no (`false`).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase CheckBox de la clase UIObject. Al acceder a estas propiedades desde el objeto CheckBox, debe utilizarse la forma *checkBoxInstance.propertyName*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Sólo lectura; anchura del objeto, expresada en píxeles.
UIObject.x	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.y	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase CheckBox de la clase UIComponent. Al acceder a estas propiedades desde el objeto CheckBox, debe utilizarse la forma *checkBoxInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase SimpleButton

En la tabla siguiente se enumeran las propiedades que hereda la clase CheckBox de la clase SimpleButton. Al acceder a estas propiedades desde el objeto CheckBox, debe utilizarse la forma *checkBoxInstance.propertyName*.

Propiedad	Descripción
SimpleButton.emphasized	Indica si el botón tiene el aspecto de un botón de comando predeterminado.
SimpleButton.emphasizedStyleDeclaration	Declaración de estilos cuando la propiedad `emphasized` está definida en `true`.

Propiedad	Descripción
SimpleButton.selected	Valor booleano que indica si el botón está seleccionado (true) o no (false). El valor predeterminado es false.
SimpleButton.toggle	Valor booleano que indica si el botón se comporta como un conmutador (true) o no (false). El valor predeterminado es false.

Propiedades heredadas de la clase Button

En la tabla siguiente se enumeran las propiedades que hereda la clase CheckBox de la clase Button. Al acceder a estas propiedades desde el objeto CheckBox, debe utilizarse la forma *checkBoxInstance.propertyName*.

Propiedad	Descripción
Button.label	Especifica el texto que aparece en un botón.
Button.labelPlacement	Especifica la orientación del texto de la etiqueta con respecto a un icono.

Resumen de eventos de la clase CheckBox

En la tabla siguiente se muestra un evento de la clase CheckBox.

Evento	Descripción
CheckBox.click	Se activa cuando se hace clic con el ratón (se suelta el botón del ratón) sobre la casilla de verificación o cuando la casilla está seleccionada y se presiona la barra espaciadora.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase CheckBox de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.

Evento	Descripción
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase CheckBox de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase SimpleButton

En la tabla siguiente se muestra el evento que hereda la clase CheckBox de la clase SimpleButton.

Evento	Descripción
SimpleButton.click	Se difunde cuando se hace clic en un botón.

CheckBox.click

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObject:Object) {
  // ...
};
checkBoxInstance.addEventListener("click", listenerObject);
```

Sintaxis 2:

```
on (click) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) sobre la casilla de verificación o cuando la casilla está seleccionada y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*checkBoxInstance*) distribuye un evento (en este caso, `click`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `addEventListener()` (véase `EventDispatcher.addEventListener()`) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de CheckBox. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la casilla de verificación `myCheckBox`, envía "_level0.myCheckBox" al panel Salida:

```
on (click) {
  trace(this);
}
```

Ejemplo

El siguiente ejemplo activa un botón cuando se selecciona la casilla de verificación. Este ejemplo presupone que hay una instancia del componente Button en el escenario con el nombre de instancia `submit_button` y una instancia del componente CheckBox en el escenario con el nombre de instancia `agree_ch`. Añada el siguiente código al primer fotograma de la línea de tiempo principal:

```
agree_ch.label = "I agree";
submit_button.enabled = false;

// Crear un objeto detector.
var form_obj:Object = new Object();

// Asignar una función al objeto detector.
form_obj.click = function(event_obj:Object) {
 submit_button.enabled = event_obj.target.selected;
};

// Añadir detector.
agree_ch.addEventListener("click", form_obj);
```

El código siguiente envía un mensaje al panel Salida cuando se hace clic en `checkBoxInstance`. El controlador `on()` debe asociarse directamente con `checkBoxInstance`:

```
on (click) {
  trace("check box component was clicked");
}
```

Véase también

`EventDispatcher.addEventListener()`

CheckBox.label

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

checkBoxInstance.label

Descripción

Propiedad; indica la etiqueta de texto de la casilla de verificación. De forma predeterminada, la etiqueta aparece a la derecha de la casilla de verificación. Si se establece esta propiedad, reemplaza el parámetro label especificado en la ficha Parámetros del inspector de componentes.

El componente CheckBox no permite etiquetas de varias líneas.

Ejemplo

El código siguiente define el texto que aparece junto al componente CheckBox y envía el valor al panel Salida:

```
checkBox.label = "Remove from list";
trace(checkBox.label)
```

Este ejemplo crea la casilla de verificación en ActionScript y, a continuación, cambia el tamaño de la etiqueta cuando la casilla se activa. Para este ejemplo, arrastre el componente CheckBox desde el panel Componentes a la biblioteca del documento actual (de modo que el componente CheckBox aparezca en la biblioteca pero no en el escenario). A continuación, añada el siguiente código ActionScript al primer fotograma de la línea de tiempo principal:

```
this.createClassObject(mx.controls.CheckBox, "my_ch", 10, {label:"Resize
  CheckBox instance"});

function checkboxHandler(evt_obj:Object):Void {
 trace("before: " + evt_obj.target.width +"px wide");
 evt_obj.target.setSize(200, evt_obj.target.height);
 trace("after: " + evt_obj.target.width+"px wide");
}
my_ch.addEventListener("click", checkboxHandler);
```

Véase también

CheckBox.labelPlacement

CheckBox.labelPlacement

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

checkBoxInstance.labelPlacement

Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a la casilla de verificación. A continuación se indican los cuatro valores posibles (las líneas discontinuas representan el área de delimitación del componente; son invisibles en el documento):

- `"right"` La casilla de verificación se fija en la esquina superior izquierda del área de delimitación. La etiqueta aparece a la derecha de la casilla. Éste es el valor predeterminado.

- `"left"` La casilla de verificación se fija en la esquina superior derecha del área de delimitación. La etiqueta aparece a la izquierda de la casilla.

- `"bottom"` La etiqueta se coloca debajo de la casilla de verificación. La casilla de verificación y la etiqueta se centran horizontal y verticalmente.

- `"top"` La etiqueta se coloca encima de la casilla de verificación. La casilla de verificación y la etiqueta se centran horizontal y verticalmente.

El área de delimitación del componente se puede cambiar durante la edición con el comando Transformar o en tiempo de ejecución con la propiedad `UIObject.setSize()`. Para más información, consulte "Personalización del componente CheckBox" en la página 139.

Ejemplo

En el ejemplo siguiente se define la posición de la etiqueta a la izquierda de la casilla de verificación:

```
checkBox_mc.labelPlacement = "left";
```

El siguiente ejemplo utiliza código ActionScript para crear instancias de casilla de verificación. La instancia de casilla de verificación `right_ch` tiene su etiqueta y sus propiedades labelPlacement definidas en el método "UIObject.createClassObject()" en la página 1404. La instancia de casilla de verificación `left_ch` tiene su etiqueta y sus propiedades labelPlacement definidas en declaraciones independientes. Arrastre el componente CheckBox desde el panel Componentes a la biblioteca del documento actual (de modo que el componente aparezca en la biblioteca pero no en el escenario). A continuación, añada el siguiente código ActionScript al primer fotograma de la línea de tiempo principal:

```
this.createClassObject(mx.controls.CheckBox, "right_ch", 1, {label:"Right",
  labelPlacement:"right"});
right_ch.move(10, 10);
this.createClassObject(mx.controls.CheckBox, "left_ch", 2);
left_ch.label= "Left";
left_ch.labelPlacement = "left";
left_ch.move(10, 30);
```

Véase también

`CheckBox.label`

CheckBox.selected

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`checkBoxInstance``.selected`

Descripción

Propiedad; valor booleano que selecciona (`true`) o anula la selección (`false`) de la casilla de verificación.

Ejemplo

El siguiente ejemplo se muestra una casilla de verificación que tiene su propiedad seleccionada definida en `true`, de forma predeterminada y, a continuación, se utiliza la propiedad `selected` en una función de controlador de eventos para responder al clic del usuario en la casilla de verificación. Arrastre un componente CheckBox al escenario. Asigne a la instancia del componente el nombre `my_ch`. A continuación, añada el siguiente código al panel Acciones del primer fotograma de la línea de tiempo principal:

```
my_ch.selected = true;

var checkboxListener:Object = new Object();
checkboxListener.click = function(evt_obj:Object) {
 if (evt_obj.target.selected) {
  evt_obj.target.label = "Selected!";
 } else {
  evt_obj.target.label = "Unselected!";
 }
};
my_ch.addEventListener("click", checkboxListener);
```

CAPÍTULO 7

Interfaz Collection (sólo en Flash Professional)

7

La clase Collection se distribuye en la biblioteca de clases común como un símbolo de clip compilado. Para acceder a esta clase, seleccione Ventana > Bibliotecas comunes > Clases, que contiene el clip compilado UtilsClasses.

Clase Collection (sólo en Flash Professional)

Nombre de clase de ActionScript mx.utils.Collection

La interfaz Collection permite administrar mediante programación un grupo de elementos relacionados, denominados *elementos de colección*. Cada elemento de colección de este conjunto tiene propiedades que se describen en los metadatos de la definición de clase del elemento de colección.

Los componentes pueden exponer propiedades como colecciones, que se pueden manipular durante la edición a través del cuadro de diálogo Valores desde el inspector de componentes. Este cuadro de diálogo permite añadir elementos, eliminar elementos, modificar propiedades de elementos y cambiar la posición de elementos en la colección. Para más información sobre colecciones y elementos de colección, consulte "Etiqueta Collection" en *Utilización de componentes*.

Generalmente se usa la interfaz Collection con componentes que usan la etiqueta de metadatos Collection para crear propiedades de colección. Aunque puede crear, acceder y eliminar instancias de Collection mediante programación, las colecciones se suelen usar en el contexto de un componente. Flash MX Professional 2004 proporciona implementaciones de las dos interfaces relacionadas con colecciones (CollectionImpl para Collection e IteratorImpl para Iterator).

Resumen de métodos de la interfaz Collection

En la tabla siguiente se enumeran los métodos de la interfaz Collection.

Método	Descripción
`Collection.addItem()`	Añade un elemento nuevo al final de la colección.
`Collection.contains()`	Indica si la colección contiene el elemento especificado.
`Collection.clear()`	Elimina todos los elementos de la colección.
`Collection.getItemAt()`	Devuelve un elemento de la colección especificado por su índice.
`Collection.getIterator()`	Devuelve un repetidor de los elementos de la colección.
`Collection.getLength()`	Devuelve el número de elementos de la colección.
`Collection.isEmpty()`	Indica si la colección está vacía.
`Collection.removeItem()`	Elimina el elemento especificado de la colección.

Collection.addItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`collection``.addItem(`*`item`*`)`

Parámetros

`item` El objeto que se va a añadir a la colección. Si el valor de *`item`* es `null`, no se añade a la colección.

Valor devuelto

Un valor booleano `true` si se modificó la colección como consecuencia de la operación.

Descripción

Método; añade un elemento nuevo al final de la colección.

Ejemplo

En el siguiente ejemplo se llama a `addItem()`:

```
on (click) {
import CompactDisc;

  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  myCD = new CompactDisc();
  myCD.Artist = "John Coltrane";
  myCD.Title = "Giant Steps";

  var wasAdded:Boolean = myColl.addItem(myCD);
}
```

Collection.contains()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`collection``.contains(`*`item`*`)`

Parámetros

`item` Objeto cuya presencia en la colección hay que probar.

Valor devuelto

Un valor booleano `true` si la colección contiene *`item`*.

Descripción

Método; indica si la colección contiene el elemento especificado. Para que Flash considere los objetos como iguales, deben hacer referencia al mismo objeto. Si *`item`* es un objeto distinto, `Collection.contains()` devuelve `false`, aunque todas las propiedades del objeto sean iguales.

Ejemplo

En el siguiente ejemplo se llama a `contains()`:

```
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;

var itr:mx.utils.Iterator = myColl.getIterator();
while (itr.hasNext()) {
  var cd:CompactDisc = CompactDisc(itr.next());
  var title:String = cd.Title;
  var artist:String = cd.Artist;

  if(myColl.contains(cd)) {
    trace("myColl contains " + title);
  }
  else {
    trace("myColl does not contain " + title);
  }
}
```

Collection.clear()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

collection`.clear()`

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la colección.

Ejemplo

En el siguiente ejemplo se llama a `clear()`:

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  myColl.clear();
}
```

Collection.getItemAt()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

collection.getItemAt(*index*)

Parámetros

index Número que indica la posición de *item* en la colección. Es un índice de base cero, por lo que 0 recupera el primer elemento, 1 recupera el segundo elemento, etc.

Valor devuelto

Un objeto que contiene una referencia al elemento de la colección especificado o `null` si el valor de *index* está fuera de los límites.

Descripción

Método; devuelve un elemento de la colección especificado por su índice.

Ejemplo

En el ejemplo siguiente se llama a `getItemAt()`:

```
//...
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  var myCD = CompactDisc(myColl.getItemAt(0));
  if (myCD !=null) {
    trace("Retrieved " + myCD.Title);
  }
//...
```

Collection.getIterator()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

collection.getIterator()

Valor devuelto

Un objeto Iterator que puede utilizar para recorrer la colección.

Descripción

Método; devuelve un repetidor de los elementos de la colección. No se sabe en qué orden se devolverán los elementos (a menos que esta colección sea una instancia de una clase que proporciona una garantía).

Ejemplo

En el ejemplo siguiente se llama a `getIterator()`:

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;

  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisc(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;

    trace("Title: " + title + " - Artist: " + artist);
  }
}
```

Collection.getLength()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

collection.getLength()

Valor devuelto

El número de elementos de la colección.

Descripción

Método; devuelve el número de elementos de la colección.

Ejemplo

En el siguiente ejemplo se llama a `getLength()`:

```
//...
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;
trace ("Collection size is: " + myColl.getLength());
//...
```

Collection.isEmpty()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

collection.isEmpty()

Valor devuelto

Un valor booleano `true` si la colección está vacía.

Descripción

Método; indica si la colección está vacía.

Ejemplo

En el ejemplo siguiente se llama a `isEmpty()`:

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  if (myColl.isEmpty()) {
    trace("No CDs in the collection");
  }
}
//...
```

Collection.removeItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`collection``.removeItem(`*`item`*`)`

Parámetros

`item` El objeto que se va a quitar de la colección.

Valor devuelto

Un valor booleano `true` si se eliminó *`item`* correctamente.

Descripción

Método; elimina el elemento especificado de la colección. Como `Collection.removeItem()` reduce dinámicamente el tamaño de la colección, no se debe llamar a este método mientras se realiza un bucle mediante un repetidor.

Ejemplo

En el siguiente ejemplo se llama a `removeItem()`:

```
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;

// obtener esto de un cuadro de introducción de texto
var removeArtist:String = _parent.tArtistToRemove.text;
var removeSize:Number = 0;

if (myColl.isEmpty()) {
  trace("No CDs in the collection");
}
else {
  var toRemove:Array = new Array();
  var itr:mx.utils.Iterator = myColl.getIterator();
  var cd:CompactDisc = new CompactDisc();
  var title:String = "";
  var artist:String = "";
  while (itr.hasNext()) {
    cd = CompactDisc(itr.next());
    title = cd.Title;
    artist = cd.Artist;
    if(artist == removeArtist) {
      // marcar este artista para su eliminación
      removeSize = toRemove.push(cd);
      trace("*** Marked for deletion: " + artist + "|" + title);
    }
  }
  // después del bucle while, eliminar los que no interesan
  var removeCD:CompactDisc = new CompactDisc();
  for(i = 0; i < removeSize; i++) {
    removeCD = toRemove[i];
    trace("Removing: " + removeCD.Artist + "|" + removeCD.Title);
    myColl.removeItem(removeCD);
  }
}
```

CAPÍTULO 8

Componente ComboBox

8

Un cuadro combinado permite al usuario hacer una sola selección en una lista emergente. Los cuadros combinados pueden ser estáticos o editables. Un cuadro combinado editable permite al usuario introducir texto directamente en un campo de texto de la parte superior de la lista y seleccionar una opción en una lista emergente. Si la lista emergente llega al final del documento, se abre hacia arriba en lugar de hacia abajo. El cuadro combinado se compone de tres subcomponentes: un componente Button, un componente TextInput y un componente List.

Cuando se hace la selección en la lista, la etiqueta de la selección se copia en el campo de texto de la parte superior del cuadro combinado. No importa si la selección se realiza con el ratón o con el teclado.

El componente ComboBox se selecciona al hacer clic en el cuadro de texto o en el botón. Cuando un componente ComboBox está seleccionado y es editable, todas las pulsaciones del teclado se dirigen al cuadro de texto y se gestionan según las reglas del componente TextInput (véase "Componente TextInput" en la página 1247), con excepción de las teclas siguientes:

Tecla	Descripción
Control+Flecha abajo	Abre la lista desplegable y la selecciona.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Cuando un componente ComboBox está seleccionado y es estático, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista desplegable al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar un cuadro combinado estático:

Tecla	Descripción
Control+Flecha abajo	Abre la lista desplegable y la selecciona.
Control+Flecha arriba	Cierra la lista desplegable si está abierta en las versiones autónoma y de navegador de Flash Player.

Tecla	Descripción
Flecha abajo	Desplaza la selección un elemento hacia abajo.
Fin	La selección se desplaza al final de la lista.
Escape	Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado en el modo de prueba.
Intro	Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado.
Inicio	Desplaza la selección a la parte superior de la lista.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Cuando la lista desplegable de un cuadro combinado está seleccionada, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar una lista desplegable:

Tecla	Descripción
Control+Flecha arriba	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y la lista desplegable se cierra en las versiones independiente y de navegador de Flash Player.
Flecha abajo	Desplaza la selección un elemento hacia abajo.
Fin	Mueve el punto de inserción al final del cuadro de texto.
Intro	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable.
Escape	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable en el modo de prueba.
Inicio	Mueve el punto de inserción al principio del cuadro de texto.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Tabulador	Desplaza la selección al objeto siguiente.
Mayús+Fin	Selecciona el texto desde el punto de inserción a la posición de fin.

Tecla	Descripción
Mayús+Inicio	Selecciona el texto desde el punto de inserción a la posición de inicio.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Flecha arriba	Desplaza la selección un elemento hacia arriba.

NOTA

Para las teclas Av Pág y Re Pág, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia del componente ComboBox del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el inspector de componentes. Sin embargo, la lista desplegable no se abre en la previsualización dinámica y el primer elemento aparece como elemento seleccionado.

Cuando se añade el componente ComboBox a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ComboBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Utilización del componente ComboBox

Utilice el componente ComboBox en formularios o aplicaciones en los que haya que elegir una sola opción en una lista. Por ejemplo, puede emplear una lista desplegable de provincias en un formulario de dirección de cliente. En supuestos más complejos, utilice cuadros combinados editables. Por ejemplo, en una aplicación de itinerarios, podría emplear un cuadro combinado editable para que el usuario introduzca las direcciones de origen y de destino. La lista desplegable contendría las direcciones introducidas con anterioridad.

Parámetros de ComboBox

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ComboBox en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

data asocia un valor de datos con cada elemento del componente ComboBox. El parámetro data es una matriz.

editable determina si el componente ComboBox es editable (`true`) o sólo seleccionable (`false`). El valor predeterminado es `false`.

labels rellena el componente ComboBox con una matriz de valores de texto.

rowCount establece el número máximo de elementos que se pueden mostrar en la lista. El valor predeterminado es 5.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente ComboBox en el inspector de componentes (Ventana > Inspector de componentes):

restrict indica el conjunto de caracteres que un usuario puede introducir en el campo de texto de un cuadro combinado. El valor predeterminado es `undefined`. Véase "ComboBox.restrict" en la página 204.

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades `minHeight` y `minWidth` se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para definir opciones adicionales para instancias de ComboBox con los métodos, propiedades y eventos de la clase ComboBox. Para más información, consulte "Clase ComboBox" en la página 174.

Creación de aplicaciones con el componente ComboBox

En el siguiente procedimiento se explica cómo añadir un componente ComboBox a una aplicación durante la edición. En este ejemplo, el cuadro combinado presenta una lista de ciudades para seleccionar en la lista emergente.

Para crear una aplicación con el componente ComboBox:

1. Arrastre un componente ComboBox desde el panel Componentes al escenario.
2. Seleccione la herramienta Transformación y cambie el tamaño del componente en el escenario.

   Sólo es posible cambiar el tamaño de un cuadro combinado en el escenario durante la edición. Normalmente sólo hace falta modificar la anchura del cuadro para adaptarla al contenido.
3. Seleccione el cuadro combinado y, en el inspector de propiedades, introduzca el nombre de instancia **comboBox**.
4. En el inspector de componentes o el inspector de propiedades, haga lo siguiente:
   - Introduzca **Minneapolis**, **Portland** y **Keene** en el parámetro label. Haga doble clic en el campo del parámetro label para abrir el cuadro de diálogo Valores. A continuación, haga clic en el signo más para añadir elementos.
   - Introduzca **MN.swf**, **OR.swf** y **NH.swf** en el parámetro data.

     Son archivos SWF imaginarios que, por ejemplo, podrían cargarse cuando el usuario seleccionara una ciudad en el cuadro combinado.
5. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
function change(evt){
  trace(evt.target.selectedItem.label);
}
comboBox.addEventListener("change", this);
```

   La última línea de código añade un controlador de eventos `change` a la instancia `ComboBox`. Para más información, consulte `ComboBox.change`.

Para crear un componente ComboBox mediante código ActionScript:

1. Arrastre el componente ComboBox desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
this.createClassObject(mx.controls.ComboBox, "my_cb", 10);

my_cb.addItem({data:1, label:"One"});
my_cb.addItem({data:2, label:"Two"});
```

Este script utiliza el método "UIObject.createClassObject()" en la página 1404 para crear la instancia de ComboBox y, a continuación, utiliza "ComboBox.addItem()" en la página 180 para añadir elementos de la lista al componente ComboBox.

3. Añada un detector de eventos y una función de controlador de eventos para responder cuando se seleccione un elemento ComboBox:

```
// Crear un objeto detector.
var cbListener:Object = new Object();
// Crear una función de controlador de eventos.
cbListener.change = function (evt_obj:Object) {
 trace("Currently selected item is: " +
   evt_obj.target.selectedItem.label);
}
// Añadir un detector de eventos.
my_cb.addEventListener("change", cbListener);
```

4. Seleccione Control > Probar película y haga clic en un elemento del componente ComboBox para ver un mensaje en el panel Salida.

Personalización del componente ComboBox

El componente ComboBox puede transformarse horizontal y verticalmente durante la edición. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar.

Si el texto es demasiado largo para encajar en el cuadro combinado, se trunca. Para encajar el texto de la etiqueta, deberá cambiar el tamaño del cuadro combinado durante la edición.

En los cuadros combinados editables, el área activa es sólo el botón, no el cuadro de texto. En los cuadros combinados estáticos, el área activa está formada por el botón y el cuadro de texto. El área activa responde abriendo o cerrando la lista desplegable.

Utilización de estilos con el componente ComboBox

Es posible definir propiedades de estilo para cambiar el aspecto de un componente ComboBox. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

Los cuadros combinados sólo tienen dos estilos exclusivos: `openDuration` y `openEasing`. Los estilos del botón, el cuadro de texto y la lista desplegable del cuadro combinado se asignan a través de dichos componentes, como se indica a continuación:

- El botón es una instancia de Button y utiliza sus estilos. (Véase “Utilización de estilos con el componente Button” en la página 96.)
- El texto es una instancia de TextInput y utiliza sus estilos. (Véase “Utilización de estilos con el componente TextInput” en la página 1250.)
- La lista desplegable es una instancia de List y utiliza sus estilos. (Véase “Utilización de estilos con el componente List” en la página 794.)

El componente ComboBox utiliza los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Ambos	Color del fondo. El color predeterminado es blanco.
`borderStyle`	Ambos	El subcomponente Button utiliza dos instancias de RectBorder para sus bordes y responde a los estilos definidos en esa clase. Véase “Clase RectBorder” en la página 1103. En el tema Halo, el componente ComboBox utiliza un borde redondeado personalizado para la parte contraída del componente ComboBox. Los colores de esta parte del componente ComboBox sólo se pueden modificar aplicando aspectos. Véase “Utilización de aspectos con el componente ComboBox” en la página 173.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).

Estilo	Tema	Descripción
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto; puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`openDuration`	Ambos	Duración, en milisegundos, de la animación de transición. El valor predeterminado es 250.
`openEasing`	Ambos	Referencia a una función de interpolación que controla la animación. De forma predeterminada es una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.

En el ejemplo siguiente se ilustra la forma de usar estilos del componente List para controlar el comportamiento de la parte emergente de un componente ComboBox.

```
// comboBox es una instancia del componente ComboBox en el escenario.
comboBox.setStyle("alternatingRowColors", [0xFFFFFF, 0xBFBFBF]);
```

Utilización de aspectos con el componente ComboBox

El componente ComboBox utiliza símbolos de la biblioteca para representar los estados del botón y tiene variables de aspecto para la flecha abajo. Estos aspectos se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/ComboBox Assets/States de los archivos HaloTheme.fla y SampleTheme.fla. La información siguiente describe estos aspectos e indica los pasos para personalizarlos.

El componente ComboBox también utiliza aspectos de barra de desplazamiento para la barra de desplazamiento de la lista desplegable y dos instancias de la clase RectBorder para el borde en torno a la entrada de texto y la lista desplegable. Para más información sobre la personalización de estos aspectos, consulte "Utilización de aspectos con el componente UIScrollBar" en la página 1435 y "Clase RectBorder" en la página 1103. Para más información sobre los métodos disponibles para aplicar aspectos a los componentes, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

El componente ComboBox utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
`ComboDownArrowDisabledName`	Estado desactivado de la flecha abajo. El valor predeterminado es `ComboDownArrowDisabled`.
`ComboDownArrowDownName`	Estado presionado de la flecha abajo. El valor predeterminado es `ComboDownArrowDown`.
`ComboDownArrowUpName`	Estado sin presionar de la flecha abajo. El valor predeterminado es `ComboDownArrowOver`.
`ComboDownArrowOverName`	Estado de la flecha abajo cuando se desplaza el puntero sobre ella. El valor predeterminado es `ComboDownArrowUp`.

Para crear símbolos de clip de película para aspectos de ComboBox:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta ComboBox Assets a la biblioteca del documento.
4. Expanda la carpeta ComboBox Assets/States en la biblioteca del documento.

5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo ComboDownArrowDisabled.

6. Personalice el símbolo como desee.

   Por ejemplo, cambie el color blanco del cuadrado interior por un gris claro.

7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente ComboBox al escenario.
10. Defina las propiedades de la instancia de ComboBox como desee.

    Para este ejemplo, use código ActionScript para establecer el estado desactivado del componente ComboBox.

11. Seleccione Control > Probar película.

Clase ComboBox

Herencia MovieClip > Clase UIObject > Clase UIComponent > ComboBase > ComboBox

Nombre de clase de ActionScript mx.controls.ComboBox

El componente ComboBox combina tres subcomponentes independientes: Button, TextInput y List. La mayoría de los métodos, propiedades y eventos de cada subcomponente están disponibles directamente desde el componente ComboBox y se muestran en las tablas de resumen para la clase ComboBox.

La lista desplegable de un cuadro combinado se obtiene con una matriz o un proveedor de datos. Si utiliza un proveedor de datos, la lista cambia en tiempo de ejecución. Puede cambiar dinámicamente el origen de los datos del ComboBox cambiando a otra matriz u otro proveedor de datos.

Los elementos de una lista de un cuadro combinado se indexan por posición a partir del número 0. Un elemento puede ser:

- Un tipo de datos simple.
- Un objeto que contiene una propiedad `label` y una propiedad `data`.

> **NOTA** Un objeto puede utilizar la propiedad `ComboBox.labelFunction` o `ComboBox.labelField` para determinar la propiedad `label`.

Si el elemento es un tipo de datos simple distinto de una cadena, se convierte en una cadena. Si el elemento es un objeto, la propiedad `label` debe ser una cadena y la propiedad `data` puede ser cualquier valor de ActionScript.

Los métodos de ComboBox a los que se suministran datos tienen dos parámetros, *label* y *data*, que hacen referencia a las propiedades anteriores. Los métodos que devuelven elementos los devuelven como objetos.

Un cuadro combinado aplaza la creación de la instancia de su lista desplegable hasta que un usuario interactúe con él. Por tanto, puede parecer que la respuesta de un cuadro combinado es lenta la primera vez que se usa.

Use el código siguiente para acceder mediante programación a la lista desplegable del componente ComboBox y sustituir la demora:

```
var foo = myComboBox.dropdown;
```

El acceso a la lista emergente puede provocar una pausa en la aplicación. Esto puede suceder cuando el usuario interactúa por primera vez con el cuadro combinado o cuando se ejecuta el código anterior.

Resumen de métodos de la clase ComboBox

En la tabla siguiente se enumeran los métodos de la clase ComboBox.

Método	Descripción
`ComboBox.addItem()`	Añade un elemento al final de la lista.
`ComboBox.addItemAt()`	Añade un elemento al final de la lista en el índice especificado.
`ComboBox.close()`	Cierra la lista desplegable.
`ComboBox.getItemAt()`	Devuelve el elemento del índice especificado.
`ComboBox.open()`	Abre la lista desplegable.
`ComboBox.removeAll()`	Elimina todos los elementos de la lista.
`ComboBox.removeItemAt()`	Elimina un elemento de la lista en la posición especificada.
`ComboBox.replaceItemAt()`	Sustituye el contenido del elemento en el índice especificado.
`ComboBox.sortItems()`	Ordena la lista mediante una función de comparación.
`ComboBox.sortItemsBy()`	Ordena la lista utilizando un campo de cada elemento.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase ComboBox de la clase UIObject. Al llamar a estos métodos desde el objeto ComboBox, debe utilizarse la forma *comboBoxInstance.methodName*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase ComboBox de la clase UIComponent. Al llamar a estos métodos desde el objeto ComboBox, debe utilizarse la forma *comboBoxInstance.methodName*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase ComboBox

En la tabla siguiente se enumeran las propiedades de la clase ComboBox.

Propiedad	Descripción
`ComboBox.dataProvider`	Modelo de datos de los elementos de la lista.
`ComboBox.dropdown`	Devuelve una referencia al componente List contenido en el cuadro combinado.
`ComboBox.dropdownWidth`	Anchura de la lista desplegable, expresada en píxeles.
`ComboBox.editable`	Indica si un cuadro combinado es editable.
`ComboBox.labelField`	Indica qué campo de datos ha de utilizarse como etiqueta para la lista desplegable.
`ComboBox.labelFunction`	Especifica una función para calcular el campo de etiqueta de la lista desplegable.
`ComboBox.length`	Sólo lectura; longitud de la lista desplegable.
`ComboBox.restrict`	El conjunto de caracteres que un usuario puede introducir en el campo de texto de un cuadro combinado.
`ComboBox.rowCount`	Número máximo de elementos de la lista que se muestran de una vez.
`ComboBox.selectedIndex`	Índice del elemento seleccionado en la lista desplegable.
`ComboBox.selectedItem`	Valor del elemento seleccionado en la lista desplegable.
`ComboBox.text`	La cadena de texto del cuadro de texto.
`ComboBox.textField`	Referencia al componente TextInput del cuadro combinado.
`ComboBox.value`	Valor del cuadro de texto (editable) o lista desplegable (estático).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase ComboBox de la clase UIObject. Al acceder a estas propiedades desde el objeto ComboBox, debe utilizarse la forma *`comboBoxInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.

Propiedad	Descripción
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase ComboBox de la clase UIComponent. Al acceder a estas propiedades desde el objeto ComboBox, debe utilizarse la forma *`comboBoxInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase ComboBox

En la tabla siguiente se enumeran los eventos de la clase ComboBox.

Evento	Descripción
`ComboBox.change`	Se difunde cuando el valor del cuadro combinado cambia como resultado de la interacción del usuario.
`ComboBox.close`	Se difunde cuando la lista del cuadro combinado empieza a replegarse.
`ComboBox.enter`	Se difunde cuando se presiona la tecla Intro.
`ComboBox.itemRollOut`	Se difunde cuando el puntero se desplaza fuera de un elemento de la lista emergente.
`ComboBox.itemRollOver`	Se difunde cuando el ratón se desplaza sobre un elemento de la lista desplegable.
`ComboBox.open`	Se difunde cuando la lista desplegable comienza a abrirse.
`ComboBox.scroll`	Se difunde cuando se recorre la lista desplegable.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase ComboBox de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase ComboBox de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

ComboBox.addItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.addItem(label[, data])
comboBoxInstance.addItem({label:label[, data:data]})
comboBoxInstance.addItem(obj);
```

Parámetros

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento; pueden ser de cualquier tipo. Este parámetro es opcional.

obj Objeto con una propiedad `label` y una propiedad `data` opcional.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista.

Ejemplo

Con una instancia del componente ComboBox denominada `my_cb`, añada el siguiente código ActionScript al panel Acciones del primer fotograma de la línea de tiempo principal. Este código ActionScript crea un ComboBox con tres elementos, cada uno de los cuales tiene un valor de datos y una cadena de etiqueta. Al probar el archivo SWF y hacer clic en uno de los elementos, el panel Salida muestra la identidad del "target", del valor de datos y de la etiqueta:

```
// Añadir elementos al Combo Box.
my_cb.addItem("this is an Item");
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

// Añadir un detector de eventos y una función de controlador de eventos.
var cbListener:Object = new Object();
cbListener.change = function(evt_obj:Object):Void  {
 var currentlySelected:Object = evt_obj.target.selectedItem;
 trace(evt_obj.target);
 trace("data: "+currentlySelected.data);
 trace("label: "+currentlySelected.label);
};
my_cb.addEventListener("change", cbListener);
```

ComboBox.addItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.addItemAt(*index*, *label*[, *data*])

comboBoxInstance.addItemAt(*index*, {label:*label*[, data:*data*]})

comboBoxInstance.addItemAt(*index*, *obj*);

Parámetros

index Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento; pueden ser de cualquier tipo. Este parámetro es opcional.

obj Objeto con propiedades `label` y `data`.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista en el índice especificado con el parámetro *index*. Los índices superiores a `ComboBox.length` se omiten.

Ejemplo

Comience con una instancia del componente ComboBox denominada `my_cb` y una instancia del componente Button denominada `my_btn`. Añada el siguiente código ActionScript al panel Acciones del primer fotograma de la línea de tiempo principal. Cuando pruebe el archivo SWF, haga clic en el cuadro combinado para ver los dos elementos que contiene. A continuación, haga clic en el botón y, cuando vuelva a hacer clic en el cuadro combinado, comprobará que se añadió otro elemento con la etiqueta "first value":

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.addItemAt(0, {data:1, label:"first value"});
};
my_btn.addEventListener("click", btnListener);
```

ComboBox.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando la propiedad `ComboBox.selectedIndex` o `ComboBox.selectedItem` cambia como consecuencia de la interacción con el usuario.

Utilizando un modelo de evento distribuidor/detector, una instancia del componente (*`comboBoxInstance`*) distribuye un evento (en este caso, `change`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama a `addEventListener()` (véase `EventDispatcher.addEventListener()`) en la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

Con una instancia del componente ComboBox `my_cb` en el escenario, el siguiente ejemplo envía el nombre de la instancia del componente que generó el evento `change` al panel Salida:

```
// Añadir elemento a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear un objeto detector.
var cbListener:Object = new Object();

// Asignar una función al objeto detector.
cbListener.change = function(event_obj:Object) {
 trace("Value changed to: "+event_obj.target.selectedItem.label);
};

// Añadir detector.
my_cb.addEventListener("change", cbListener);
```

Véase también

`EventDispatcher.addEventListener()`

ComboBox.close()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.close()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; cierra la lista desplegable.

Ejemplo

Con una instancia del componente ComboBox denominada `my_cb` en el escenario y una instancia del componente Button denominada `my_button`, el siguiente ejemplo cierra la lista desplegable del cuadro combinado `my_cb` cuando se hace clic en el botón `my_button` :

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.close();
};
my_button.addEventListener("click", btnListener);
```

Véase también

ComboBox.open()

ComboBox.close

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("close", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando la lista desplegable del cuadro combinado se repliega completamente.

Utilizando un modelo de evento distribuidor/detector, una instancia del componente (*comboBoxInstance*) distribuye un evento (en este caso, `close`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

Con una instancia `my_cb` del componente ComboBox en el escenario, el siguiente ejemplo envía un mensaje al panel Salida cuando se abre o se cierra la lista desplegable:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear un objeto detector.
var cbListener:Object = new Object();
cbListener.open = function(evt_obj:Object) {
 trace("The ComboBox has opened.");
}
cbListener.close = function(evt_obj:Object){
   trace("The ComboBox has closed.");
}

// Añadir detector.
my_cb.addEventListener("open", cbListener);
my_cb.addEventListener("close", cbListener);

// Abrir el cuadro combinado.
my_cb.open();
```

Véase también

`EventDispatcher.addEventListener()`

ComboBox.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`comboBoxInstance``.dataProvider`

Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la API DataProvider. El valor predeterminado es `[]`. El componente List y el componente ComboBox comparten la propiedad `dataProvider`; los cambios de esta propiedad están disponibles inmediatamente para ambos componentes.

El componente List, como otros componentes para datos, añade métodos al prototipo del objeto Array para ceñirse a la API DataProvider (para más detalles, consulte DataProvider.as). Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, se accede a las propiedades `labelField` o `labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es `"label"`, por lo que, si existe este campo, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

NOTA	Si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar, se pierde la selección.

Como proveedor de datos para un componente List pueden elegirse todas las instancias que implementen la API DataProvider. Esto incluye los objetos RecordSet de Flash Remoting, los componentes DataSet de Firefly, etc.

Ejemplo

Este ejemplo utiliza una matriz de cadenas para rellenar la lista desplegable de la instancia del componente ComboBox `my_cb`:

```
my_cb.dataProvider = [{data:1, label:"First Item"}, {data:2, label:"Second
  Item"}];
/* es lo mismo que
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
*/
```

En este ejemplo se crea una matriz de proveedor de datos y se asigna dicha matriz a la propiedad `dataProvider`:

```
var myDP:Array = new Array();
list.dataProvider = myDP;

for (var i:Number = 0; i < accounts.length; i++) {
  // Estos cambios en DataProvider se difundirán a la lista.
  myDP.addItem({label: accounts[i].name,
                data:  accounts[i].accountID});
}
```

ComboBox.dropdown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.dropdown

Descripción

Propiedad (sólo lectura); devuelve una referencia a la lista contenida en el cuadro combinado. No se crea una instancia en el cuadro combinado para el subcomponente List hasta que se tenga que visualizar. No obstante, cuando se accede a la propiedad `dropdown`, se crea la lista.

Ejemplo

Con una instancia del componente ComboBox my_cb en el escenario y dos símbolos de clip de película en la biblioteca con valores de identificador de vinculación definidos en `dw_id` y `fl_id`, el siguiente código ActionScript utiliza la propiedad dropdown para añadir iconos a cada elemento de la lista desplegable:

```
// Definir la anchura de la lista desplegable para adaptar los tamaños de la
  etiqueta.
my_cb.dropdownWidth = 200;

// Definir el estilo de iconField en la propiedad dropdown de ComboBox.
// La propiedad dropdown es una referencia para el componente List en el
  ComboBox
// para poder definir estilos de List para ComboBox.
my_cb.dropdown.setStyle("iconField", "pIcon");

// Añadir elementos a la lista.
my_cb.addItem({label:"Dreamweaver 1", pIcon:"dw_id"});
my_cb.addItem({label:"Flash 1", pIcon:"fl_id"});
my_cb.addItem({label:"Flash 2", pIcon:"fl_id"});
```

Véase también

`ComboBox.dropdownWidth`

ComboBox.dropdownWidth

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.dropdownWidth

Descripción

Propiedad; límite de anchura de la lista desplegable, expresado en píxeles. El valor predeterminado es la anchura del componente ComboBox (instancia de TextInput más instancia de SimpleButton).

Ejemplo

Con una instancia del componente ComboBox `my_cb` en el escenario, el siguiente código ActionScript define la anchura de la lista desplegable para que quepan las etiquetas:

```
// Definir la anchura de la lista desplegable para adaptar los tamaños de la
  etiqueta.
my_cb.dropdownWidth = 200;

// Añadir elementos a la lista.
my_cb.addItem("ComboBox");
my_cb.addItem({data:2, label:"This is a long label"});
my_cb.addItem({data:3, label:"This has an even longer label"});
```

Véase también

ComboBox.dropdown

ComboBox.editable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.editable

Descripción

Propiedad; indica si el cuadro combinado es editable (`true`) o no (`false`). En un cuadro combinado editable, un usuario puede introducir en el cuadro de texto valores que no aparecen en la lista desplegable. Si el cuadro combinado no es editable, no se puede introducir texto en el cuadro de texto. El cuadro de texto muestra el texto del elemento de la lista. El valor predeterminado es `false`.

Si se convierte un cuadro combinado en editable, se borra el campo de texto del cuadro combinado. Además, se establece el índice (y el elemento) seleccionado en `undefined`. Para convertir un cuadro combinado en editable conservando el elemento seleccionado, utilice el código siguiente:

```
var ix:Number = myComboBox.selectedIndex;
myComboBox.editable = true; // Borra el campo de texto.
myComboBox.selectedIndex = ix; // Vuelve a copiar la etiqueta en el campo de
  texto.
```

Ejemplo

Con una instancia del componente ComboBox `my_cb` en el escenario, el siguiente código ActionScript crea una lista de cuadro combinado y dos detectores. El primer detector controla las acciones del ratón (o clics) en la etiqueta "Add new item" para permitir la edición en el campo del cuadro combinado. El segundo detector controla cuándo el usuario presiona la tecla Intro para añadir su entrada a la lista del cuadro combinado:

```
// Añadir elementos a la lista del cuadro combinado.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:-1, label:"Add new item..."});

// Responder al usuario que hace clic en "Add new item".
function changeListener(evt_obj:Object) {
 if (evt_obj.target.selectedItem.data == -1) {
  evt_obj.target.editable = true;
 } else if (evt_obj.target.selectedIndex != undefined) {
  evt_obj.target.editable = false;
  evt_obj.target.setFocus();
 }
}
my_cb.addEventListener("change", changeListener);

// Responder al usuario que presiona la tecla Intro después de añadir un
  nuevo nombre de elemento.
function enterListener(evt_obj:Object) {
 if (evt_obj.target.value != '') {
  evt_obj.target.addItem({data:'', label:evt_obj.target.value});
 }
 evt_obj.target.editable = false;
 evt_obj.target.selectedIndex = evt_obj.target.dataProvider.length-1;
 evt_obj.target.setFocus();
}
my_cb.addEventListener("enter", enterListener);
```

ComboBox.enter

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.enter = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("enter", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el usuario presiona la tecla Intro en el cuadro de texto. Este evento es un evento de TextInput que sólo se difunde desde cuadros combinados editables. Para más información, consulte `TextInput.enter`.

Utilizando un modelo de evento distribuidor/detector, una instancia del componente (*`comboBoxInstance`*) distribuye un evento (en este caso, `enter`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

Con una instancia del componente ComboBox `my_cb` en el escenario, el siguiente código ActionScript crea una lista de cuadro combinado y dos detectores. El primer detector controla las acciones del ratón (o clics) en la etiqueta “Add new item” para permitir la edición en el campo del cuadro combinado. El segundo detector controla cuándo el usuario presiona la tecla Intro para añadir su entrada a la lista del cuadro combinado:

```
// Añadir elementos a la lista del cuadro combinado.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:-1, label:"Add new item..."});
```

```
// Responder al usuario que hace clic en "Add new item".
function changeListener(evt_obj:Object) {
 if (evt_obj.target.selectedItem.data == -1) {
  evt_obj.target.editable = true;
 } else if (evt_obj.target.selectedIndex != undefined) {
  evt_obj.target.editable = false;
  evt_obj.target.setFocus();
 }
}
my_cb.addEventListener("change", changeListener);

// Responder al usuario que presiona la tecla Intro después de añadir un
   nuevo nombre de elemento.
function enterListener(evt_obj:Object) {
 if (evt_obj.target.value != '') {
  evt_obj.target.addItem({data:'', label:evt_obj.target.value});
 }
 evt_obj.target.editable = false;
 evt_obj.target.selectedIndex = evt_obj.target.dataProvider.length-1;
 evt_obj.target.setFocus();
}
my_cb.addEventListener("enter", enterListener);
```

Véase también

`EventDispatcher.addEventListener()`

ComboBox.getItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.getItemAt(*index*)

Parámetros

index El índice del elemento que se va a recuperar. El índice debe ser un número mayor o igual que 0, y menor que el valor de `ComboBox.length`.

Valor devuelto

Valor u objeto del elemento indexado. Si el índice está fuera de rango, el valor es undefined.

Descripción

Método; recupera el elemento del índice especificado.

Ejemplo

Con una instancia `my_cb` del componente ComboBox en el escenario, el siguiente código ActionScript muestra la etiqueta del primer elemento del cuadro combinado en el panel Salida:

```
// Añadir elemento a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

trace(my_cb.getItemAt(1).label);
```

ComboBox.itemRollOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.itemRollOut = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("itemRollOut", listenerObject)
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `itemRollOut` tiene una propiedad `index` adicional: El índice es el número del elemento desde el que se desplazó el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero deja de estar sobre los elementos de la lista emergente. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte `List.itemRollOut`.

Utilizando un modelo de evento distribuidor/detector, una instancia del componente (*comboBoxInstance*) distribuye un evento (en este caso, `itemRollOut`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

Con una instancia de ComboBox `my_cb` en el escenario, el siguiente código ActionScript envía un mensaje al panel Salida que indica el índice de elemento y el evento cuando el puntero del ratón se desplaza o deja de desplazarse por encima de un elemento:

```
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear un objeto detector.
var cbListener:Object = new Object();
cbListener.itemRollOver = function(evt_obj:Object) {
 trace("index: "+evt_obj.index+", event: "+evt_obj.type);
};
cbListener.itemRollOut = function(evt_obj:Object) {
 trace("index: "+evt_obj.index+", event: "+evt_obj.type);
};

// Añadir detector.
my_cb.addEventListener("itemRollOver", cbListener);
my_cb.addEventListener("itemRollOut", cbListener);
```

Véase también

`ComboBox.itemRollOver`, `EventDispatcher.addEventListener()`

ComboBox.itemRollOver

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.itemRollOver = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("itemRollOver", listenerObject)
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `itemRollOver` tiene una propiedad `index` adicional: El índice es el número del elemento sobre el que se desplazó el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza sobre los elementos de la lista emergente. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte `List.itemRollOver`.

Utilizando el modelo de evento distribuidor/detector, una instancia del componente (*`comboBoxInstance`*) distribuye un evento (en este caso, `itemRollOver`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

Con una instancia de ComboBox `my_cb` en el escenario, el siguiente código ActionScript envía un mensaje al panel Salida que indica el índice de elemento y el evento cuando el puntero del ratón se desplaza o deja de desplazarse por encima de un elemento:

```
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear un objeto detector.
var cbListener:Object = new Object();
cbListener.itemRollOver = function(evt_obj:Object) {
 trace("index: " + evt_obj.index + ", event: " + evt_obj.type);
};
cbListener.itemRollOut = function(evt_obj:Object) {
 trace("index: " + evt_obj.index + ", event: " + evt_obj.type);
};

// Añadir detector.
my_cb.addEventListener("itemRollOver", cbListener);
my_cb.addEventListener("itemRollOut", cbListener);
```

Véase también

`ComboBox.itemRollOut`, `EventDispatcher.addEventListener()`

ComboBox.labelField

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`comboBoxInstance``.labelField`

Descripción

Propiedad; nombre del campo de los objetos de matriz `dataProvider` que se utilizará como campo de etiqueta. Ésta es una propiedad del componente List que está disponible desde una instancia del componente ComboBox. Para más información, consulte `List.labelField`.

El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se define la propiedad `dataProvider` en una matriz de cadenas y la propiedad `labelField` para que indique que el campo `name` debe utilizarse como etiqueta de la lista desplegable:

```
my_cb.dataProvider = [
  {name:"Gary", gender:"male"},
  {name:"Susan", gender:"female"} ];

my_cb.labelField = "name";
```

Véase también

`List.labelFunction`

ComboBox.labelFunction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`comboBoxInstance``.labelFunction`

Descripción

Propiedad; función que calcula la etiqueta de un elemento proveedor de datos. Es preciso definir la función. El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se crea un proveedor de datos y después se define una función que especifica qué debe utilizarse como etiqueta en la lista desplegable:

```
myComboBox.dataProvider  = [
  {firstName:"Nigel", lastName:"Pegg", age:"really young"},
  {firstName:"Gary", lastName:"Grossman", age:"young"},
  {firstName:"Chris", lastName:"Walcott", age:"old"},
  {firstName:"Greg", lastName:"Yachuk", age:"really old"} ];

myComboBox.labelFunction = function(itemObj){
  return (itemObj.lastName + ", " + itemObj.firstName);
}
```

Véase también

`List.labelField`

ComboBox.length

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.length
```

Descripción

Propiedad (sólo lectura); longitud de la lista desplegable. Ésta es una propiedad del componente List que está disponible desde una instancia del componente ComboBox. Para más información, consulte `List.length`. El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se almacena el valor `length` en una variable:

```
var dropdownItemCount:Number = myComboBox.length;
```

ComboBox.open()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.open()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; abre la lista desplegable.

Ejemplo

Con una instancia del componente ComboBox denominada `my_cb` en el escenario y una instancia del componente Button denominada `my_button`, el siguiente ejemplo abre la lista desplegable del cuadro combinado `my_cb` cuando se hace clic en el botón `my_button`:

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.open();
};
my_button.addEventListener("click", btnListener);
```

Véase también

`ComboBox.close()`

ComboBox.open

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.open = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("open", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando la lista desplegable se abre completamente.

Utilizando el modelo de evento distribuidor/detector, una instancia del componente (*comboBoxInstance*) distribuye un evento (en este caso, `open`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

Con una instancia `my_cb` del componente ComboBox en el escenario, el siguiente ejemplo envía un mensaje al panel Salida cuando se abre o se cierra la lista desplegable:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear un objeto detector.
var cbListener:Object = new Object();
cbListener.open = function(evt_obj:Object) {
 trace("The ComboBox has opened.");
}
cbListener.close = function(evt_obj:Object){
   trace("The ComboBox has closed.");
}

// Añadir detector.
my_cb.addEventListener("open", cbListener);
my_cb.addEventListener("close", cbListener);

// Abrir el cuadro combinado.
my_cb.open();
```

Véase también

`ComboBox.close`, `EventDispatcher.addEventListener()`

ComboBox.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.removeAll()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la lista. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

Ejemplo

Con una instancia de ComboBox `my_cb` en el escenario y una instancia del componente Button `clear_button` en el escenario, el siguiente código ActionScript coloca el cuadro combinado y el botón uno junto al otro. Cuando haga clic en el cuadro combinado, verá una lista de elementos. Cuando haga clic en el botón, se borrarán los elementos del cuadro combinado:

```
my_cb.move(10, 10);
clear_button.move(120, 10);

// Crear proveedor de datos.
var myDP_array:Array = new Array();
myDP_array.push({data:1, label:"First Item"});
myDP_array.push({data:2, label:"Second Item"});

my_cb.dataProvider = myDP_array;

// Definir un objeto de detector de eventos.
var clearListener:Object = new Object();
clearListener.click = function(evt_obj:Object){
 my_cb.removeAll();
}

// Añadir detector.
clear_button.addEventListener("click", clearListener);
```

Véase también

ComboBox.removeItemAt(), ComboBox.replaceItemAt()

ComboBox.removeItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.removeItemAt(*index*)

Parámetros

index Número que indica la posición del elemento que va a eliminarse. El índice tiene base cero.

Valor devuelto

Un objeto; el elemento eliminado (undefined si no hay elementos).

Descripción

Método; elimina el elemento de la posición de índice especificada. Los índices de la lista a partir del índice que indica el parámetro *index* se contraen una posición. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

Ejemplo

Con una instancia de ComboBox `my_cb` en el escenario y una instancia del componente Button `clear_button` en el escenario, el siguiente código ActionScript coloca el cuadro combinado y el botón uno junto al otro. Cuando haga clic en el cuadro combinado, verá una lista de dos elementos. Cuando haga clic en el botón, se borrará el segundo elemento del cuadro combinado (en la posición de índice `1`, porque el valor está basado en cero):

```
my_cb.move(10, 10);
clear_button.move(120, 10);

// Crear proveedor de datos.
var myDP_array:Array = new Array();
myDP_array.push({data:1, label:"First Item"});
myDP_array.push({data:2, label:"Second Item"});

my_cb.dataProvider = myDP_array;
```

```
// Definir un objeto de detector de eventos.
var clearListener:Object = new Object();
clearListener.click = function(evt_obj:Object){
 my_cb.removeItemAt(1);
}

// Añadir detector.
clear_button.addEventListener("click", clearListener);
```

Véase también

`ComboBox.removeAll()`, `ComboBox.replaceItemAt()`

ComboBox.replaceItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.replaceItemAt(*index*, *label*[, *data*])

comboBoxInstance.replaceItemAt(*index*, {label:*label*[, data:*data*]})

comboBoxInstance.replaceItemAt(*index*, *obj*);

Parámetros

index Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es opcional.

obj Objeto con propiedades `label` y `data`.

Valor devuelto

Ninguno.

Descripción

Método; sustituye el contenido del elemento en el índice especificado. Éste es un método del componente List que está disponible desde el componente ComboBox.

Ejemplo

Con una instancia del componente ComboBox `my_cb` y una instancia del componente TextInput `label_ti` en el escenario, el siguiente código ActionScript añade la entrada de usuario al cuadro combinado cuando el usuario presiona la tecla Intro:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Crear detector de la acción del usuario al presionar la tecla Intro en el
  campo de entrada de texto.
var tiListener:Object = new Object();
tiListener.enter = function(evt_obj:Object) {
 my_cb.replaceItemAt(my_cb.selectedIndex, {label:evt_obj.target.text});
 // Es necesario para actualizar la entrada de ComboBox modificada
  recientemente
 my_cb.selectedIndex = my_cb.selectedIndex;
};
label_ti.addEventListener("enter", tiListener);
```

Véase también

`ComboBox.removeAll()`, `ComboBox.removeItemAt()`

ComboBox.restrict

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`comboBoxInstance``.restrict`

Descripción

Propiedad; indica el conjunto de caracteres que un usuario puede introducir en el campo de texto de un cuadro combinado. El valor predeterminado es `undefined`. Si el valor de esta propiedad es `null` o una cadena vacía (`""`), el usuario puede introducir cualquier carácter. Si esta propiedad contiene una cadena de caracteres, el usuario sólo puede introducir caracteres en la cadena; la cadena se explora de izquierda a derecha. Puede especificarse un rango utilizando un guión (`-`).

Si la cadena empieza con un símbolo de intercalación (^), todos los caracteres que sigan al símbolo de intercalación se considerarán caracteres no válidos. Si la cadena no empieza por un símbolo de intercalación, se aceptan los caracteres de la cadena.

Puede utilizar la barra diagonal inversa (\) para introducir un guión (-), el símbolo de intercalación (^) o el carácter de barra diagonal inversa (\), como se muestra a continuación:

```
\^
\-
\\
```

Cuando se introduce una barra diagonal inversa en el panel Acciones entre comillas dobles, tiene un significado especial para el intérprete de comillas dobles del panel Acciones. Significa que el carácter que va a continuación de la barra diagonal debe interpretarse "literalmente". Por ejemplo, se podría utilizar el siguiente código para introducir una comilla sencilla:

```
var leftQuote = "\'";
```

El intérprete restrict del panel Acciones también utiliza la barra diagonal inversa como carácter de escape. Por tanto, es posible que crea que la siguiente expresión debe funcionar:

```
myText.restrict = "0-9\-\^\\";
```

Sin embargo, como esta expresión está escrita entre comillas dobles, se envía el valor `0-9-^\` al intérprete restrict, que no comprenderá este valor.

Debe escribir esta expresión entre comillas dobles; no sólo debe enviarla al intérprete restrict, sino que también debe escribir una barra diagonal inversa antes de las comillas dobles en el intérprete incorporado del panel Acciones, para interpretarlas literalmente. Para enviar el valor `0-9\-\^\\` al intérprete restrict, debe introducir el siguiente código:

```
myCombo.restrict = "0-9\\-\\^\\\\";
```

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación Incorporar contornos de fuentes del inspector de propiedades.

Ejemplo

Con una instancia del componente ComboBox `my_cb`, el siguiente código ActionScript restringe la entrada de caracteres a números de 0 a 9, guiones y puntos:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Permitir la edición del cuadro combinado.
my_cb.editable = true;

// Restringir los caracteres que pueden introducirse en el cuadro combinado.
my_cb.restrict = "0-9\\-\\.\\ ";
```

En el ejemplo siguiente, la primera línea del código limita el campo de texto a letras en mayúsculas, números y espacios. La segunda línea del código admite todos los caracteres excepto letras en minúsculas.

```
my_combo.restrict = "A-Z 0-9";
my_combo.restrict = "^a-z";
```

El código siguiente permite al usuario introducir los caracteres "0 1 2 3 4 5 6 7 8 9 - ^ \" en la instancia `myCombo`. Debe utilizar dos barras diagonales inversas para interpretar literalmente los caracteres `-`, `^` y `\`. El primer carácter `\` interpreta literalmente las comillas dobles y el segundo carácter `\` indica al intérprete que el carácter que va a continuación no debe interpretarse como un carácter especial.

```
myCombo.restrict = "0-9\\-\\^\\\\";
```

ComboBox.rowCount

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`comboBoxInstance``.rowCount`

Descripción

Propiedad; el número máximo de filas visible en la lista desplegable antes de que el cuadro combinado inserte una barra de desplazamiento. El valor predeterminado es 5.

Si el número de elementos de la lista desplegable es mayor que el valor de la propiedad `rowCount`, se ajustará el tamaño de la lista y se mostrará una barra de desplazamiento si es necesario. Si la lista desplegable contiene menos elementos que la propiedad `rowCount`, adapta su tamaño al número de elementos de la lista.

Este comportamiento difiere del componente List, que siempre muestra el número de filas que especifique la propiedad `rowCount` aunque quede espacio vacío.

Si el valor es negativo o fraccionario, el comportamiento es indefinido.

Ejemplo

Con una instancia del componente ComboBox `my_cb`, el siguiente código ActionScript hace que el cuadro combinado muestre los tres primeros elementos y añade una barra de desplazamiento para ver el cuarto:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Mostrar barra de desplazamiento si ComboBox tiene más de 3 elementos.
my_cb.rowCount = 3;
```

ComboBox.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // El código se escribe aquí.
};
comboBoxInstance.addEventListener("scroll", listenerObject);
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `scroll` tiene una propiedad adicional, direction. Es una cadena con dos valores posibles, `"horizontal"` o `"vertical"`. En los eventos `scroll` de ComboBox, el valor siempre es `"vertical"`.

Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista desplegable. Éste es un evento del componente List disponible para el componente ComboBox.

Utilizando un modelo de evento distribuidor/detector, una instancia del componente (*comboBoxInstance*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

Con una instancia del componente ComboBox `my_cb`, en el ejemplo siguiente se envía un mensaje al panel Salida que indica el índice del elemento al que se desplaza la lista:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Mostrar barra de desplazamiento si ComboBox tiene más de 2 elementos.
my_cb.rowCount = 3;

// Crear un objeto detector.
var cbListener:Object = new Object();
cbListener.scroll = function(evt_obj:Object) {
 trace("The list had been scrolled to item # "+evt_obj.position);
};

// Añadir detector.
my_cb.addEventListener("scroll", cbListener);
```

Véase también

`EventDispatcher.addEventListener()`

ComboBox.selectedIndex

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.selectedIndex

Descripción

Propiedad; el número de índice del elemento seleccionado en la lista desplegable. El valor predeterminado es 0. Cuando se asigna esta propiedad, se borra la selección actual, se selecciona el elemento indicado y el cuadro de texto del cuadro combinado muestra la etiqueta del elemento indicado.

Si asigna un valor fuera de rango a esta propiedad, Flash lo omitirá. Cuando se introduce texto en el campo de texto de un cuadro combinado editable, `selectedIndex` se establece en `undefined`.

Ejemplo

Con una instancia del componente ComboBox `my_cb`, el siguiente código selecciona el último elemento de la lista (si no, aparecería de forma predeterminada el primer elemento):

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Seleccionar el último elemento de la lista.
my_cb.selectedIndex = my_cb.length-1;
```

Véase también

`ComboBox.selectedItem`

ComboBox.selectedItem

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.selectedItem

Descripción

Propiedad; valor del elemento seleccionado en la lista desplegable.

Si el cuadro combinado es editable, `selectedItem` devolverá `undefined` cuando el usuario introduzca texto en el cuadro de texto. Esta propiedad sólo tiene un valor si se selecciona un elemento de la lista desplegable o se establece mediante código ActionScript. Si el cuadro combinado es estático, el valor de `selectedItem` siempre es válido; devuelve `undefined` si no hay elementos en la lista.

Ejemplo

Con una instancia del componente ComboBox `my_cb`, el siguiente ejemplo muestra los valores para las propiedades de datos y de etiqueta `selectedItem`:

```
// Añadir elementos a la lista.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

var cbListener:Object = new Object();
cbListener.change = function(evt_obj:Object) {
 var item_obj:Object = my_cb.selectedItem;
 var i:String;
 for (i in item_obj) {
  trace(i + ":\t" + item_obj[i]);
 }
 trace("");
};
my_cb.addEventListener("change", cbListener);
```

Véase también

`ComboBox.dataProvider`, `ComboBox.selectedIndex`

ComboBox.sortItems()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
comboBoxInstance.sortItems([compareFunc], [optionsFlag])
```

Parámetros

compareFunc Referencia a una función que se utiliza para comparar dos elementos y determinar su orden de clasificación. Para ver más detalles, consulte `Array.sort()` en *Referencia del lenguaje ActionScript 2.0*. Este parámetro es opcional.

optionsFlag Permite realizar varios tipos de ordenación distintos en una matriz individual sin tener que replicar la matriz completa ni volver a ordenarla repetidamente. Este parámetro es opcional.

Los valores posibles de *optionsFlag* son:

- `Array.DESCENDING`, que ordena de mayor a menor.
- `Array.CASEINSENSITIVE`, que ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`, que ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas, que se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado esa etiqueta.
- `Array.UNIQUESORT`, que devuelve un código de error (0) en lugar de una matriz ordenada si dos objetos de la matriz son idénticos o lo son sus campos de ordenación.
- `Array.RETURNINDEXEDARRAY`, que devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, la siguiente matriz devolvería la segunda línea de código y la matriz no cambiaría:

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos del cuadro combinado según la función de comparación o las opciones de ordenación especificadas.

Ejemplo

En este ejemplo la ordenación se basa en las etiquetas en mayúsculas. Los elementos `a` y `b` se pasan a la función y contienen campos `label` y `data`:

```
myComboBox.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

En el ejemplo siguiente se usa la función `upperCaseFunc()` (definida arriba) y el parámetro *`optionsFlag`* para ordenar los elementos de una instancia de ComboBox denominada `myComboBox`:

```
myComboBox.addItem("Mercury");
myComboBox.addItem("Venus");
myComboBox.addItem("Earth");
myComboBox.addItem("planet");
myComboBox.sortItems(upperCaseFunc, Array.DESCENDING);
// La ordenación de myComboBox resultante será:
// Venus
// planet
// Mercury
// Earth
```

ComboBox.sortItemsBy()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

`comboBoxInstance``.sortItemsBy(`*`fieldName`*`, `*`order`*` [`*`optionsFlag`*`])`

Parámetros

fieldName Cadena que especifica el nombre del campo que se va a utilizar para la ordenación. Normalmente, este valor es `"label"` o `"data"`.

order Cadena que especifica si los elementos deben clasificarse en orden ascendente (`"ASC"`) o descendente (`"DESC"`).

optionsFlag Permite realizar varios tipos de ordenación distintos en una matriz individual sin tener que replicar la matriz completa ni volver a ordenarla repetidamente. Este parámetro es opcional pero, si se usa, debe reemplazar al parámetro *order*.

Los valores posibles de *optionsFlag* son:

- `Array.DESCENDING`, que ordena de mayor a menor.
- `Array.CASEINSENSITIVE`, que ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`, que ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas, que se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado esa etiqueta.
- `Array.UNIQUESORT`, que devuelve un código de error (0) en lugar de una matriz ordenada si dos objetos de la matriz son idénticos o lo son sus campos de ordenación.
- `Array.RETURNINDEXEDARRAY`, que devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, la siguiente matriz devolvería la segunda línea de código y la matriz no cambiaría:

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos del cuadro combinado alfabética o numéricamente y en el orden especificado mediante el nombre de campo especificado. Si los elementos de *fieldName* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *fieldName* suele ser `"label"` o `"data"`, pero los programadores avanzados pueden especificar cualquier valor primitivo. Si lo desea, puede utilizar el parámetro *optionsFlag* para especificar un estilo de ordenación.

Ejemplo

Los ejemplos siguientes se basan en una instancia de ComboBox denominada `myComboBox`, que contiene cuatro elementos denominados `"Apples"`, `"Bananas"`, `"cherries"` y `"Grapes"`:

```
// En primer lugar, llene el componente ComboBox con los elementos.
myComboBox.addItem("Bananas");
myComboBox.addItem("Apples");
myComboBox.addItem("cherries");
myComboBox.addItem("Grapes");

// La sentencia siguiente ordena con el parámetro order establecido en "ASC"
// y produce una ordenación que coloca "cherries" al final de la lista
// porque la ordenación distingue mayúsculas de minúsculas.
myComboBox.sortItemsBy("label", "ASC");
// orden resultante: Apples, Bananas, Grapes, cherries

// La sentencia siguiente ordena con el parámetro order establecido en
  "DESC"
// y produce una ordenación que coloca "cherries" al principio de la lista
// porque la ordenación distingue mayúsculas de minúsculas.
myComboBox.sortItemsBy("label", "DESC");
// orden resultante: cherries, Grapes, Bananas, Apples

// La sentencia siguiente ordena con el parámetro optionFlag establecido en
  "ASC"
// Array.CASEINSENSITIVE. Observe que el valor predeterminado es un orden
  ascendente.
myComboBox.sortItemsBy("label", Array.CASEINSENSITIVE);
// orden resultante: Apples, Bananas, cherries, Grapes

// La sentencia siguiente ordena con el parámetro optionFlag establecido en
  "ASC"
// Array.CASEINSENSITIVE | Array.DESCENDING.
myComboBox.sortItemsBy("label", Array.CASEINSENSITIVE | Array.DESCENDING);
// orden resultante: Grapes, cherries, Bananas, Apples
```

ComboBox.text

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.text

Descripción

Propiedad; texto del cuadro de texto. Este valor se puede obtener y definir con cuadros combinados editables. Con cuadros combinados estáticos, el valor es de sólo lectura.

Ejemplo

En el ejemplo siguiente se define el valor actual de `text` de un cuadro combinado editable:

```
my_cb.addItem("Arkansas");
my_cb.addItem("Georgia");

my_cb.editable = true;
my_cb.text = "California";
```

ComboBox.textField

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.textField

Descripción

Propiedad (sólo lectura); referencia al componente TextInput contenido en el componente ComboBox.

Esta propiedad permite acceder al componente TextInput subyacente para manipularlo. Por ejemplo, quizá desee cambiar la selección del cuadro de texto o limitar los caracteres que se pueden introducir en él.

Ejemplo

El código siguiente restringe el cuadro de texto de `myComboBox` para que sólo acepte números hasta un máximo de seis caracteres:

```
// Añadir elementos a la lista.
my_cb.addItem({data:0xFFFFFF, label:"white"});
my_cb.addItem({data:0x000000, label:"black"});

my_cb.editable = true;

// Restringir lo que se puede introducir en el campo de texto a 0-9.
my_cb.restrict = "0-9";

// Limitar entradas a 6 caracteres.
my_cb.textField.maxChars = 6;
```

ComboBox.value

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.value

Descripción

Propiedad de sólo lectura; si el cuadro combinado es editable, `value` devuelve la etiqueta del elemento. Si el cuadro combinado es estático, `value` devuelve los datos del elemento.

Ejemplo

En el ejemplo siguiente se introducen los datos en el cuadro combinado al definir la propiedad `dataProvider`. Después muestra `value` en el panel Salida. Por último, se selecciona `"California"` y se muestra en el panel Salida mientras el cuadro combinado sea editable; después, se muestra `"CA"` cuando el cuadro combinado ya no es editable.

```
my_cb.dataProvider = [
  {label:"Alaska", data:"AK"},
  {label:"California", data:"CA"},
  {label:"Washington", data:"WA"}];
my_cb.editable = true;
my_cb.selectedIndex = 1;
trace('Editable value is "California": ' + my_cb.value);
my_cb.editable = false;
my_cb.selectedIndex = 1;
trace('Non-editable value is "CA": ' + my_cb.value);
```

CAPÍTULO 9

Clases de vinculación de datos (sólo en Flash Professional)

9

Las clases de vinculación de datos proporcionan la funcionalidad en tiempo de ejecución para la función de vinculación de datos en Flash Professional 8. Puede crear y configurar visualmente vinculaciones de datos en el entorno de edición de Flash mediante la ficha Vinculaciones del inspector de componentes, así como crear y configurar mediante programación vinculaciones utilizando las clases incluidas en el paquete mx.data.binding.

Para ver información general sobre vinculación de datos y sobre cómo crear visualmente vinculaciones de datos en la herramienta de edición Flash, consulte "Vinculación de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Disponibilidad de las clases de vinculación de datos en tiempo de ejecución (sólo en Flash Professional)

Para compilar el archivo SWF, la biblioteca debe contener archivos SWC con código de bytes para las clases de vinculación de datos y las clases de servicio Web. Si crea vinculaciones de datos en Flash durante la edición, se añadirán automáticamente a la biblioteca las clases de componentes relevantes. Si trabaja con vinculación de datos y servicios Web en tiempo de ejecución, debe añadir las clases a la biblioteca del archivo FLA. Puede obtener estos archivos SWC de la biblioteca común Clases.

Para añadir los archivos SWC a su biblioteca:

1. Seleccione la biblioteca Clases (Ventana > Bibliotecas comunes > Clases).
2. Abra la biblioteca del documento (Ventana > Biblioteca).
3. Arrastre los archivos SWC apropiados (DataBindingClasses, WebServiceClasses o ambos) desde la biblioteca Clases a la biblioteca del documento.

Para más información sobre estas clases, consulte "Clase Binding (sólo en Flash Professional)" en la página 218 y "Clases de servicios Web (sólo en Flash Professional)" en la página 1455.

Clases del paquete mx.data.binding (sólo en Flash Professional)

En la tabla siguiente se enumeran las clases incluidas en el paquete mx.data.binding:

Clase	Descripción
Clase Binding (sólo en Flash Professional)	Crea una vinculación entre dos puntos finales.
Clase ComponentMixins (sólo en Flash Professional)	Añade funcionalidad de vinculación de datos a los componentes.
Clase CustomFormatter (sólo en Flash Professional)	Clase base para crear clases de formateador personalizado.
Clase CustomValidator (sólo en Flash Professional)	Clase base para crear clases de validador personalizado.
Clase DataType (sólo en Flash Professional)	Ofrece un acceso de lectura y escritura a los campos de datos de una propiedad de componente.
Clase EndPoint (sólo en Flash Professional)	Define el origen o el destino de una vinculación.
Clase TypedValue (sólo en Flash Professional)	Contiene un valor de datos e información sobre el tipo de datos del valor.

Clase Binding (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.Binding

La clase Binding define una asociación entre dos puntos finales: un origen y un destino. Detecta cambios en el punto final de origen y copia los datos modificados en el punto final de destino cada vez que el origen cambia.

Puede escribir vinculaciones personalizadas mediante la clase Binding (y las clases de soporte) o usar la ficha Vinculaciones del inspector de componentes.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA. Para más información, consulte **"**Disponibilidad de las clases de vinculación de datos en tiempo de ejecución (sólo en Flash Professional)**" en la página 217**.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de métodos de la clase Binding

En la tabla siguiente se enumeran los métodos de la clase Binding.

Método	Descripción
`Binding.execute()`	Recoge los datos del componente de origen, les da formato y los asigna al componente de destino.

Constructor para la clase Binding

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
new Binding(source, destination, [format], [isTwoWay])
```

Parámetros

source Punto final de origen de la vinculación. Este parámetro es nominalmente de tipo mx.data.binding.EndPoint, pero puede ser cualquier objeto ActionScript que cuente con los campos EndPoint necesarios (véase "Clase EndPoint (sólo en Flash Professional)" en la página 231).

destination Punto final de destino de la vinculación. Este parámetro es nominalmente de tipo `mx.data.binding.EndPoint`, pero puede ser cualquier objeto ActionScript que cuente con los campos EndPoint necesarios.

format Objeto opcional que contiene información de formato. El objeto debe tener las propiedades siguientes:

- `cls` Clase ActionScript que amplía la clase mx.data.binding.DataAccessor.
- `settings` Objeto cuyas propiedades proporcionan una configuración opcional para la clase de formateador especificada por `cls`.

isTwoWay Valor booleano opcional que especifica si el nuevo objeto Binding es bidireccional (`true`) o no lo es (`false`). El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Constructor; crea un objeto Binding. Puede vincular datos a cualquier objeto ActionScript que tenga propiedades y emita eventos como los componentes (entre otros).

Un objeto de vinculación existirá mientras el clip de película más interior contenga los componentes de origen y de destino. Por ejemplo, si un clip de película denominado A contiene los componentes X e Y, y existe una vinculación entre X e Y, la vinculación existirá mientras exista el clip de película A.

NOTA	No es necesario conservar una referencia al nuevo objeto Binding. En cuanto se crea, el objeto Binding empieza inmediatamente a detectar eventos modificados emitidos por cada punto final. Sin embargo, en algunos casos deberá guardar una referencia al nuevo objeto Binding para poder llamar a su método `execute()` más adelante (véase `Binding.execute()`).

Ejemplo

En este ejemplo, la propiedad `text` de un componente TextInput (`src_txt`) está vinculada a la propiedad `text` de otro componente TextInput (`dest_txt`). Cuando el campo de texto `src_txt` deja de estar seleccionado (es decir, cuando se genera el evento `focusOut`), el valor de su propiedad `text` se copia en `dest_txt.text`.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest = new EndPoint();
dest.component = dest_txt;
dest.property = "text";

new Binding(src, dest);
```

En este ejemplo se muestra cómo crear un objeto Binding que utiliza una clase personalizada de formateador. Para más información, consulte “Clase CustomFormatter (sólo en Flash Professional)” en la página 222.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest = new EndPoint();
dest.component = text_dest;
dest.property = "text";

new Binding(src, dest, {cls: mx.data.formatters.Custom, settings:
  {classname: "com.mycompany.SpecialFormatter"}});
```

Binding.execute()

Disponibilidad

Flash Player 6.

Edición

Flash MX Professional 2004.

Utilización

myBinding.execute([*reverse*])

Parámetros

reverse Valor booleano que especifica si la vinculación debe ejecutarse también desde el destino al origen (`true`) o sólo desde el origen al destino (`false`). El valor predeterminado es `false`.

Valor devuelto

Valor `null` si la vinculación se ejecutó correctamente; de lo contrario, el método devuelve una matriz de cadenas de mensajes de error que describen los errores que impidieron la ejecución de la vinculación.

Descripción

Método; recoge los datos del componente de origen y los asigna al componente de destino. Si la vinculación utiliza un formateador, se da formato a los datos antes de asignarlos al destino.

Este método también valida los datos y provoca un evento `valid` o `invalid` que deben emitir los componentes de origen y de destino. Los datos se asignan al destino aunque no sea válido, a no ser que el destino sea de sólo lectura.

Si el valor del parámetro *reverse* es `true` y la vinculación es bidireccional, ésta se ejecuta a la inversa (desde el destino al origen).

Ejemplo

El código siguiente, asociado con una instancia de componente Button, ejecuta la vinculación a la inversa (desde el componente de destino al componente de origen) cuando se hace clic en el botón.

```
on(click) {
  _root.myBinding.execute(true);
}
```

Clase CustomFormatter (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.CustomFormatter

La clase CustomFormatter define dos métodos, `format()` y `unformat()`, que proporcionan la capacidad de transformar valores de datos de un tipo de datos determinado en una cadena y viceversa. De forma predeterminada, estos métodos no realizan ninguna función. Debe implementarlos en una subclase de mx.data.binding.CustomFormatter.

Para crear su propio formateador personalizado, primero se ha de crear una subclase de CustomFormatter que implemente los métodos `format()` y `unformat()`. A continuación, asigne esta clase a una vinculación entre componentes mediante la creación de un objeto Binding con ActionScript (véase "Clase Binding (sólo en Flash Professional)" en la página 218) o a través de la ficha Vinculaciones del inspector de componentes. Para obtener información sobre cómo asignar una clase de formateador mediante el inspector de componentes, consulte "Formateadores de esquema" en *Utilización de Flash*.

También puede asignar una clase de formateador a una propiedad de componente en la ficha Esquema del inspector de componentes. En este caso, sin embargo, el formateador sólo se utilizará cuando sea necesario aplicar a los datos formato de cadena. En cambio, los formateadores que se asignan mediante el panel Vinculaciones o que se crean con ActionScript se utilizan siempre que se ejecuta la vinculación.

Para ver un ejemplo en el que se muestra cómo escribir y asignar un formateador personalizado mediante ActionScript, consulte "Ejemplo de formateador personalizado" en la página 223.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Ejemplo de formateador personalizado

En el ejemplo siguiente se muestra cómo crear una clase de formateador personalizado y aplicarla a una vinculación entre dos componentes mediante ActionScript. En este ejemplo, el valor actual de un componente NumericStepper (su propiedad `value`) está vinculado al valor actual de un componente TextInput (su propiedad `text`). La clase de formateador personalizado da formato al valor numérico actual del componente NumericStepper (por ejemplo, 1, 2 o 3) como su correspondiente en palabras en inglés (por ejemplo, “one”, “two” o “three”) antes de asignarlo al componente TextInput.

Para crear y utilizar un formateador personalizado:

1. En Flash, cree un nuevo archivo ActionScript.
2. Añada el siguiente código al archivo:

```
// NumberFormatter.as
class NumberFormatter extends mx.data.binding.CustomFormatter {
  // Aplicar formato a un valor numérico, devolver una cadena
  function format(rawValue) {
    var returnValue;
    var strArray = new Array('one', 'two', 'three');
    var numArray = new Array(1, 2, 3);
    returnValue = 0;
    for (var i = 0; i<strArray.length; i++) {
      if (rawValue == numArray[i]) {
        returnValue = strArray[i];
        break;
      }
    }
    return returnValue;
  } // convertir un valor con formato, devolver un valor sin formato
  function unformat(formattedValue) {
    var returnValue;
    var strArray = new Array('one', 'two', 'three');
    var numArray = new Array(1, 2, 3);
    returnValue = "invalid";
    for (var i = 0; i<strArray.length; i++) {
      if (formattedValue == strArray[i]) {
        returnValue = numArray[i];
        break;
      }
    }
    return returnValue;
  }
}
```

3. Guarde el archivo ActionScript con el nombre NumberFormatter.as.
4. Cree un archivo de Flash (FLA).

5. Arrastre un componente TextInput desde el panel Componentes al escenario y asígnele el nombre **textInput**. A continuación, arrastre un componente NumericStepper al escenario y asígnele el nombre **stepper**.
6. Abra la línea de tiempo y seleccione el primer fotograma de la capa 1.
7. En el panel Acciones, añada el código siguiente:

```
import mx.data.binding.*;
var x:NumberFormatter;
var customBinding = new Binding({component:stepper, property:"value",
  event:"change"}, {component:textInput, property:"text",
  event:"enter,change"}, {cls:mx.data.formatters.Custom,
  settings:{classname:"NumberFormatter"}});
```

   La segunda línea de código (`var x:NumberFormatter`) garantiza que el código de bytes de la clase de formateador personalizado se incluya en el archivo SWF compilado.
8. Seleccione Ventana > Bibliotecas comunes > Clases para abrir la biblioteca Clases.
9. Seleccione Ventana > Biblioteca para abrir la biblioteca del documento.
10. Arrastre DataBindingClasses desde la biblioteca Clases a la biblioteca del documento.

    De este modo, las clases en tiempo de ejecución de vinculación de datos están disponibles para el documento.
11. Guarde el archivo FLA en la misma carpeta que contiene NumberFormatter.as.
12. Pruebe el archivo (Control > Probar película).

    Haga clic en los botones del componente NumericStepper y observe el contenido de la actualización del componente TextInput.

Resumen de métodos de la clase CustomFormatter

En la tabla siguiente se enumeran los métodos de la clase CustomFormatter.

Método	Descripción
`CustomFormatter.format()`	Convierte un tipo de datos sin formato en un nuevo objeto.
`CustomFormatter.unformat()`	Convierte de una cadena u otro tipo de datos a un tipo de datos sin formato.

CustomFormatter.format()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

Este método se llama de forma automática, no se debe invocar directamente.

Parámetros

rawData Datos a los que se va a asignar formato.

Valor devuelto

Un valor con formato.

Descripción

Método; convierte un tipo de datos sin formato en un nuevo objeto.

Este método no se implementa de forma predeterminada. Debe definirlo en la subclase de mx.data.binding.CustomFormatter.

Para más información, consulte "Ejemplo de formateador personalizado" en la página 223.

CustomFormatter.unformat()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

Este método se llama de forma automática, no se debe invocar directamente.

Parámetros

formattedData Datos con formato que deben reconvertirse en un tipo de datos sin formato.

Valor devuelto

Un valor sin formato.

Descripción

Método; convierte una cadena, u otro tipo de datos, en un tipo de datos sin formato. Esta transformación debe ser la transformación inversa exacta de `CustomFormatter.format()`.

Este método no se implementa de forma predeterminada. Debe definirlo en la subclase de mx.data.binding.CustomFormatter.

Para más información, consulte "Ejemplo de formateador personalizado" en la página 223.

Clase CustomValidator (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.CustomValidator

La clase CustomValidator se utiliza para realizar una validación personalizada de un campo de datos contenido en un componente.

Para crear una clase de validador personalizada, primero debe crearse una subclase de mx.data.binding.CustomValidator que implemente un método denominado `validate()`. Este método recibe automáticamente un valor para validarlo. Para más información sobre cómo implementar este método, consulte `CustomValidator.validate()`.

A continuación, debe asignarse una clase de validador personalizada a un campo de un componente mediante la ficha Esquema del inspector de componentes. Para ver un ejemplo de creación y utilización de una clase de validador personalizada, consulte el apartado Ejemplo en la entrada de `CustomValidator.validate()`.

Para asignar un validador personalizado:

1. En el inspector de componentes, haga clic en la ficha Esquema.
2. Seleccione el campo que desea validar y, a continuación, seleccione Personalizado en el menú emergente Tipo de datos.
3. Seleccione el campo Opciones de validación (en la parte inferior de la ficha Esquema) y haga clic en el icono de lupa para abrir el cuadro de diálogo Configuración de validación personalizada.

4. En el cuadro de texto Clase de ActionScript, especifique el nombre de la clase de validador personalizada que ha creado.

   Para que la clase que ha especificado se incluya en el archivo SWF publicado, debe guardarla en la ruta de clases.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de métodos de la clase CustomValidator

En la tabla siguiente se enumeran los métodos de la clase CustomValidator.

Método	Descripción
`CustomValidator.validate()`	Lleva a cabo la validación de los datos.
`CustomValidator.validationError()`	Notifica errores de validación.

CustomValidator.validate()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

Este método se llama de forma automática, no se debe invocar directamente.

Parámetros

value Datos que se van a validar. Pueden ser de cualquier tipo.

Valor devuelto

Ninguno.

Descripción

Método; se llama de forma automática para validar los datos que contiene el parámetro *value*. Debe implementar este método en la subclase de CustomValidator; la implementación predeterminada no realiza ninguna acción.

Puede utilizar código ActionScript para examinar y validar los datos. Si los datos no son válidos, este método debe llamar a `this.validationError()` con el mensaje pertinente. Puede llamar a `this.validationError()` más de una vez si se detectan varios problemas de validación de los datos.

Como se puede llamar a `validate()` repetidamente, evite añadir código que tarde mucho tiempo en ejecutarse. La implementación de este método sólo debe comprobar la validación y notificar cualquier error mediante `CustomValidator.validationError()`. Asimismo, la implementación no debe realizar ninguna acción como resultado de la prueba de validación, como por ejemplo advertir al usuario final. En su lugar, cree detectores de eventos para los eventos `valid` e `invalid` y advierta al usuario final desde estos detectores de eventos (consulte el ejemplo que se muestra a continuación).

Ejemplo

En el siguiente procedimiento se muestra cómo crear y utilizar una clase de validador personalizada. El método `validate()` de la clase CustomValidator OddNumbersOnly.as clasifica como no válido cualquier valor que no sea un número impar. La validación se realiza siempre que se produzca un cambio en el valor de un componente NumericStepper vinculado a la propiedad `text` de un componente Label.

Para crear y utilizar una clase de validador personalizada:

1. En Flash, cree un nuevo archivo ActionScript (AS).
2. Añada el siguiente código al archivo AS:

```
class OddNumbersOnly extends mx.data.binding.CustomValidator
{
  public function validate(value) {
    // comprobar que el valor es de tipo Number
    var n = Number(value);
    if (String(n) == "NaN") {
      this.validationError("'" + value + "' is not a number.");
      return;
    }
    // comprobar que el número es impar
    if (n % 2 == 0) {
      this.validationError("'" + value + "' is not an odd number.");
      return;
    }
    // los datos son correctos, no debe realizarse ninguna acción, sólo
  devolver
  }
}
```

3. Guarde el archivo AS como OddNumbersOnly.as.

NOTA	El nombre del archivo AS debe coincidir con el nombre de la clase.

4. Cree un archivo de Flash (FLA).
5. Abra el panel Componentes.
6. Arrastre un componente NumericStepper desde el panel Componentes al escenario y asígnele el nombre **stepper**.
7. Arrastre un componente Label al escenario y asígnele el nombre **textLabel**.
8. Arrastre un componente TextArea al escenario y denomínelo **status**.
9. Seleccione el componente NumericStepper y abra el inspector de componentes.
10. Seleccione la ficha Vinculaciones en el inspector de componentes y haga clic en el botón Añadir vinculación (+).
11. Seleccione la propiedad Value (la única) en el cuadro de diálogo Añadir vinculaciones y haga clic en Aceptar.
12. En el inspector de componentes, haga doble clic en Vinculado a en el panel Atributos de vinculación de la ficha Vinculaciones para abrir el cuadro de diálogo Vinculado a.
13. En el cuadro de diálogo Vinculado a, seleccione el componente Label del panel Ruta del componente y su propiedad `text` en el panel Ubicación del esquema. Haga clic en Aceptar.
14. Seleccione el componente Label del escenario y haga clic en la ficha Esquema del inspector de componentes.
15. En el panel Atributos de esquema, seleccione Personalizado en el menú emergente de tipo de datos.
16. Haga doble clic en el campo Opciones de validación del panel Atributos de esquema para abrir el cuadro de diálogo Configuración de validación personalizada.
17. En el cuadro de texto Clase de ActionScript, especifique **OddNumbersOnly**, que es el nombre de la clase de ActionScript que ha creado anteriormente. Haga clic en Aceptar.
18. Abra la línea de tiempo y seleccione el primer fotograma de la capa 1.
19. Abra el panel Acciones.

20. Añada el código siguiente al panel Acciones:

```
function dataIsInvalid(evt) {
  if (evt.property == "text") {
    status.text = evt.messages;
  }
}

function dataIsValid(evt) {
  if (evt.property == "text") {
    status.text = "OK";
  }
}

textLabel.addEventListener("valid", dataIsValid);
textLabel.addEventListener("invalid", dataIsInvalid);
```

21. Guarde el archivo FLA como OddOnly.fla en la misma carpeta que contiene OddNumbersOnly.as.

22. Pruebe el archivo SWF (Control > Probar película).

Haga clic en las flechas del componente NumericStepper para modificar su valor. Observe el mensaje que aparece en el componente TextArea cuando selecciona números pares e impares.

CustomValidator.validationError()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`this.validationError(`*`errorMessage`*`)`

NOTA	Este método sólo puede invocarse desde una clase de validador personalizada; la palabra clave `this` hace referencia al objeto CustomValidator actual.

Parámetros

`errorMessage` Cadena que contiene el mensaje de error que debe notificarse.

Valor devuelto

Ninguno.

Descripción

Método; se llama desde el método `validate()` de la subclase de CustomValidator para notificar los errores de validación. Si no se llama a `validationError()`, se genera un evento `valid` cuando finalice la ejecución de `validate()`. Si se llama a `validationError()` una o más veces desde `validate()`, se genera un evento `invalid` cuando finalice la ejecución de `validate()`.

Todos los mensajes que se pasan a `validationError()` están disponibles en la propiedad `messages` del objeto de evento que se ha pasado al controlador de eventos `invalid`.

Ejemplo

Véase el apartado Ejemplo de `CustomValidator.validate()`.

Clase EndPoint (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.EndPoint

La clase EndPoint define el origen o el destino de una vinculación. Los objetos EndPoint definen un valor constante, una propiedad de componente o un campo concreto de una propiedad de componente desde los que se pueden obtener datos o a los que se pueden asignar datos. También pueden definir un evento, o una lista de eventos, que detecta un objeto Binding; cuando el evento especificado tiene lugar, se ejecuta la vinculación.

Cuando se crea una vinculación con el constructor de la clase Binding, se pasan dos objetos EndPoint: uno para el origen y uno para el destino.

```
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Los objetos EndPoint, `srcEndPoint` y `destEndPoint` pueden definirse de la manera siguiente:

```
var srcEndPoint = new mx.data.binding.EndPoint();
var destEndPoint = new mx.data.binding.EndPoint();
srcEndPoint.component = source_txt;
srcEndPoint.property = "text";
srcEndPoint.event = "focusOut";
destEndPoint.component = dest_txt;
destEndPoint.property = "text";
```

El código anterior significa que “cuando el texto de origen deja de estar seleccionado, debe copiarse el valor de su propiedad `text` en la propiedad `text` del campo de texto de destino”.

También puede pasar objetos ActionScript genéricos al constructor Binding, en lugar de pasar objetos EndPoint construidos explícitamente. El único requisito es que los objetos definan las propiedades de EndPoint necesarias, concretamente `component` y `property`. El código siguiente es equivalente al mostrado anteriormente.

```
var srcEndPoint = {component:source_txt, property:"text"};
var destEndPoint = {component:dest_txt, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de propiedades de la clase EndPoint

En la tabla siguiente se enumeran las propiedades de la clase EndPoint.

Método	Descripción
`EndPoint.component`	Referencia a una instancia de componente.
`EndPoint.constant`	Valor constante.
`EndPoint.event`	Nombre del evento o matriz de nombres de evento que emitirá el componente cuando los datos cambien.
`EndPoint.location`	Ubicación de un campo de datos dentro de la propiedad de la instancia de componente.
`EndPoint.property`	Nombre de una propiedad de la instancia de componente especificada por `EndPoint.component`.

Constructor de la clase EndPoint

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
new EndPoint()
```

Valor devuelto

Ninguno.

Descripción

Constructor; crea un objeto EndPoint.

Ejemplo

En este ejemplo se crea un objeto EndPoint denominado `source_obj` y se asignan valores a sus propiedades `component` y `property`:

```
var source_obj = new mx.data.binding.EndPoint();
source_obj.component = myTextField;
source_obj.property = "text";
```

EndPoint.component

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

endPointObj.component

Descripción

Propiedad; referencia a una instancia de componente.

Ejemplo

En este ejemplo se asigna una instancia del componente List (`listBox1`) como parámetro de componente de un objeto EndPoint.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.component = listBox1;
```

EndPoint.constant

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

endPoint_src.constant

Descripción

Propiedad; valor constante asignado a un objeto EndPoint. Esta propiedad sólo se puede aplicar a objetos EndPoint que sean el origen, y no el destino, de una vinculación entre componentes. El valor puede ser cualquier tipo de datos compatible con el destino de la vinculación. Si se especifica esta propiedad, se omitirán todas las demás propiedades de EndPoint para el objeto EndPoint especificado.

Ejemplo

En este ejemplo, se asigna el valor constante de tipo cadena "hello" a la propiedad constant de un objeto EndPoint:

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.constant = "hello";
```

EndPoint.event

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

endPointObj.event

Descripción

Propiedad; especifica el nombre de un evento, o una matriz de nombres de evento, generado por el componente cuando se modifican los datos asignados a la propiedad vinculada. Cuando tiene lugar un evento, se ejecuta la vinculación.

El evento especificado sólo se aplica a los componentes que se utilizan como origen de una vinculación o como destino de una vinculación bidireccional. Para más información sobre la creación de vinculaciones bidireccionales, consulte "Clase Binding (sólo en Flash Professional)" en la página 218.

Ejemplo

En este ejemplo, la propiedad `text` de un componente TextInput (`src_txt`) se vincula a la misma propiedad de otro componente TextInput (`dest_txt`). La vinculación se ejecuta cuando el componente `src_txt` emite el evento `focusOut` o `enter`.

```
var source = {component:src_txt, property:"text", event:["focusOut",
  "enter"]};
var dest = {component:myTextArea, property:"text"};
var newBind = new mx.data.binding.Binding(source, dest);
```

EndPoint.location

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

endPointObj`.location`

Descripción

Propiedad; especifica la ubicación de un campo de datos en la propiedad de la instancia de componente. Existen cuatro maneras de especificar una ubicación: como una cadena que contiene una expresión XPath, como una cadena que contiene una ruta de ActionScript, como una matriz de cadenas o como un objeto.

Las expresiones XPath sólo se pueden utilizar cuando los datos son un objeto XML. Véase el ejemplo 1 que aparece más adelante. Para ver una lista de expresiones XPath admitidas, consulte "Creación de vinculaciones utilizando expresiones de ruta" en *Utilización de Flash*.

Para los objetos XML y ActionScript, también puede especificar una cadena que contenga una ruta ActionScript. Una ruta ActionScript contiene los nombres de los campos separados por puntos (por ejemplo, `"a.b.c"`).

También puede especificar una matriz de cadenas como ubicación. Cada cadena de la matriz pasa a otro nivel de anidación. Puede utilizar esta técnica tanto con los datos XML como con los datos ActionScript. Véase el ejemplo 2 que aparece más adelante. Cuando se utiliza con datos ActionScript, una matriz de cadenas es equivalente a utilizar código ActionScript; es decir, la matriz `["a","b","c"]` es equivalente a `"a.b.c"`.

Si especifica un objeto como ubicación, el objeto debe especificar dos propiedades: `path` e `indices`. La propiedad `path` es una matriz de cadenas, como se ha indicado anteriormente, excepto que una o varias de las cadenas especificadas puede ser el símbolo especial `"[n]"`. Para cada instancia de este símbolo en `path`, debe haber el elemento de índice correspondiente en `indices`. Mientras se evalúa la ruta, los índices se utilizan para indexar las matrices. El elemento de índice puede ser cualquier objeto EndPoint. Este tipo de ubicación sólo se puede aplicar a los datos ActionScript, no a XML. Véase el ejemplo 3 que aparece más adelante.

Ejemplo

Ejemplo 1: en este ejemplo se utiliza una expresión XPath para especificar la ubicación de un nodo denominado `zip` en un objeto XML:

```
var sourceEndPoint = new mx.databinding.EndPoint();
var sourcObj = new Object();
sourceObj.xml = new XML("<zip>94103</zip>");
sourceEndPoint.component = sourceObj;
sourceEndPoint.property = "xml";
sourceEndPoint.location = "/zip";
```

Ejemplo 2: en este ejemplo se utiliza una matriz de cadenas para pasar a una propiedad de un clip de película anidado:

```
var sourceEndPoint = new mx.data.binding.EndPoint();
// Presuponer que existe movieClip1.ball.position.
sourceEndPoint.component = movieClip1;
sourceEndPoint.property = "ball";
// Acceder a movieClip1.ball.position.x.
sourceEndPoint.location = ["position","x"];
```

Ejemplo 3: en este ejemplo se muestra cómo utilizar un objeto para especificar la ubicación de un campo de datos en una estructura de datos compleja:

```
var city = new Object();
city.theaters = [{theater: "t1", movies: [{name: "Good,Bad,Ugly"},
  {name:"Matrix Reloaded"}]}, {theater: "t2", movies: [{name: "Gladiator"},
  {name: "Catch me if you can"}]}];
var srcEndPoint = new EndPoint();
srcEndPoint.component = city;
srcEndPoint.property = "theaters";
srcEndPoint.location = {path: ["[n]","movies","[n]","name"], indices:
  [{constant:0},{constant:0}]};
```

EndPoint.property

Disponibilidad

Flash Player 6 (6.0.79.0)

Edición

Flash MX Professional 2004.

Utilización

endPointObj.property

Descripción

Propiedad; especifica un nombre de propiedad de la instancia de componente especificada por `EndPoint.component` que contiene los datos que pueden vincularse.

NOTA	`EndPoint.component` y `EndPoint.property` deben combinarse para formar una combinación válida de objeto/propiedad de ActionScript.

Ejemplo

En este ejemplo se vincula la propiedad `text` de un componente TextInput (`text_1`) a la misma propiedad de otro componente TextInput (`text_2`).

```
var sourceEndPoint = {component:text_1, property:"text"};
var destEndPoint = {component:text_2, property:"text"};
new Binding(sourceEndPoint, destEndPoint);
```

Clase ComponentMixins (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.ComponentMixins

La clase ComponentMixins define las propiedades y los métodos que se añaden automáticamente a cualquier objeto que sea origen o destino de una vinculación o a cualquier componente que sea destino de una llamada al método `ComponentMixins.initComponent()`. Estas propiedades y estos métodos no afectan a las funciones normales de los componentes, sino que añaden funciones útiles para la vinculación de datos.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de métodos de la clase ComponentMixins

En la tabla siguiente se enumeran los métodos de la clase ComponentMixins.

Método	Descripción
`ComponentMixins.getField()`	Devuelve un objeto para obtener y configurar el valor de un campo en una ubicación específica de una propiedad de componente.
`ComponentMixins.initComponent()`	Añade los métodos ComponentMixins a un componente.
`ComponentMixins.refreshDestinations()`	Ejecuta todas las vinculaciones que tengan a este objeto como punto final de origen.
`ComponentMixins.refreshFromSources()`	Ejecuta todas las vinculaciones que tengan a este componente como punto final de destino.
`ComponentMixins.validateProperty()`	Comprueba si los datos de la propiedad indicada son válidos.

ComponentMixins.getField()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

componentInstance.getField(*propertyName*, [*location*])

Parámetros

propertyName Cadena que contiene el nombre de una propiedad del componente especificado.

location Parámetro opcional que indica la ubicación de un campo en la propiedad de componente. Resulta útil cuando *propertyName* especifica una estructura de datos compleja y le interesa un campo concreto de esa estructura. La propiedad *location* puede tener las tres formas siguientes:

- Una cadena que contiene una expresión XPath. Esto sólo es válido para estructuras de datos XML. Para ver una lista de expresiones XPath admitidas, consulte "Creación de vinculaciones utilizando expresiones de ruta" en *Utilización de Flash*.

- Una cadena que contiene nombres de campo, separados por puntos (por ejemplo `"a.b.c"`). Esta forma puede utilizarse con datos complejos (ActionScript o XML).
- Una matriz de cadenas, en la que cada cadena es un nombre de campo (por ejemplo `["a", "b", "c"]`). Esta forma puede utilizarse con datos complejos (ActionScript o XML).

Valor devuelto

Un objeto DataType.

Descripción

Método; devuelve un objeto DataType cuyos métodos pueden utilizarse para obtener o establecer el valor de datos en la propiedad de componente en la ubicación de campo especificada. Para más información, consulte "Clase DataType (sólo en Flash Professional)" en la página 244.

Ejemplo

Este ejemplo utiliza el método `DataType.setAsString()` para establecer el valor de un campo ubicado en la propiedad de un componente. En este caso, la propiedad (`results`) es una estructura de datos complejos.

```
import mx.data.binding.*;
var field : DataType = myComponent.getField("results", "po.address.name1");
field.setAsString("Teri Randall");
```

Véase también

`DataType.setAsString()`

ComponentMixins.initComponent()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mx.data.binding.ComponentMixins.initComponent(`*`componentInstance`*`)`

Parámetros

`componentInstance` Referencia a una instancia de componente.

Valor devuelto

Ninguno.

Descripción

Método (estático); añade todos los métodos ComponentMixins al componente especificado por *componentInstance*. Se llama automáticamente a este método para todos los componentes implicados en una vinculación de datos. Para que los métodos ComponentMixins estén disponibles para un componente no implicado en una vinculación de datos, debe llamar explícitamente a este método para dicho componente.

Ejemplo

El código siguiente hace que los métodos ComponentMixins estén disponibles para un componente DataSet:

```
mx.data.binding.ComponentMixins.initComponent(_root.myDataSet);
```

ComponentMixins.refreshDestinations()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

componentInstance.refreshDestinations()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta todas las vinculaciones para las que *componentInstance* es el objeto EndPoint de origen. Este método permite ejecutar vinculaciones cuyas fuentes no emiten ningún evento de modificación de datos.

Ejemplo

En el ejemplo siguiente se ejecutan todas las vinculaciones para las que la instancia del componente DataSet denominada `user_data` es el objeto EndPoint de origen:

```
user_data.refreshDestinations();
```

ComponentMixins.refreshFromSources()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

componentInstance.refreshFromSources()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta todas las vinculaciones para las que *componentInstance* es el objeto EndPoint de destino. Este método permite ejecutar vinculaciones que tienen fuentes constantes o fuentes que no emiten ningún evento de modificación de datos.

Ejemplo

El ejemplo siguiente ejecuta todas las vinculaciones para las que la instancia de componente ListBox denominada `cityList` es el objeto EndPoint de destino:

```
cityList.refreshFromSources();
```

ComponentMixins.validateProperty()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

componentInstance.validateProperty(*propertyName*)

Parámetros

propertyName Cadena que contiene el nombre de una propiedad que pertenece a *componentInstance*.

Valor devuelto

Una matriz o el valor `null`.

Descripción

Método; determina si los datos de *propertyName* son válidos según la configuración de esquema de la propiedad. La configuración de esquema de la propiedad es la que se especifica en la ficha Esquema del inspector de componentes.

El método devuelve el valor `null` si los datos son válidos; de lo contrario, devuelve una matriz de mensajes de error en forma de cadenas.

La validación sólo se aplica a los campos cuya información de esquema se encuentra disponible. Si un campo es un objeto que contiene otros campos, se validará cada campo "secundario" de forma recursiva. Cada campo individual distribuirá un evento `valid` o `invalid`, según sea necesario. Para cada campo de datos que contenga *propertyName*, este método distribuirá eventos `valid` o `invalid` de la forma siguiente:

- Si el valor del campo es `null` y *no* es obligatorio, el método devolverá el valor `null`. No se genera ningún evento.
- Si el valor del campo es `null` y *es* obligatorio, se devolverá un error y se generará un evento `invalid`.
- Si el valor del campo no es `null` y el esquema del campo *no* tiene un validador, el método devuelve `null`; no se genera ningún evento.
- Si el valor no es `null` y el esquema del campo *define* un validador, el objeto validador procesa los datos. Si los datos son válidos, se generará un evento `valid` y se devolverá el valor `null`; de lo contrario, se generará un evento `invalid` y se devolverá una matriz de cadenas de error.

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar `validateProperty()` para garantizar que la longitud del texto introducido por el usuario sea válida. La longitud válida se determina configurando las Opciones de validación para el tipo de datos String en la ficha Esquema del inspector de componentes. Si el usuario introduce en el campo de texto una cadena cuya longitud no es válida, se muestran en el panel Salida los mensajes de error devueltos por `validateProperty()`.

Para validar el texto introducido por un usuario en un componente TextInput:

1. Arrastre un componente TextInput desde el panel Componentes al escenario y asígnele el nombre **zipCode_txt**.
2. Seleccione el componente TextInput y, en el inspector de componentes, haga clic en la ficha Esquema.
3. En el panel Árbol de esquema (panel superior de la ficha Esquema) seleccione la propiedad `text`.
4. En el panel Atributos de esquema (panel inferior de la ficha Esquema), seleccione ZipCode en el menú emergente de tipo de datos.
5. Si la línea de tiempo no está abierta, ábrala.
6. Haga clic en el primer fotograma de la Capa 1 de la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
7. Añada el código siguiente al panel Acciones:

```
// Añadir métodos ComponentMixin al componente TextInput.
// Tenga en cuenta que es necesario realizar este paso sólo si el
  componente
// todavía no forma parte de una vinculación de datos,
// ya sea como origen o destino.
mx.data.binding.ComponentMixins.initComponent(zipCode_txt);
// Definir la función de detector de eventos para el componente:
validateResults = function (eventObj) {
  var errors:Array = eventObj.target.validateProperty("text");
  if (errors != null) {
    trace(errors);
  }
};
// Registrar la función de detector con el componente:
zipCode_txt.addEventListener("enter", validateResults);
```

8. Seleccione Ventana > Bibliotecas comunes > Clases para abrir la biblioteca Clases.
9. Seleccione Ventana > Biblioteca para abrir la biblioteca del documento.
10. Arrastre DataBindingClasses desde la biblioteca Clases a la biblioteca del documento.

    Este paso hace que las clases de vinculación de datos en tiempo de ejecución estén disponibles para el archivo SWF en tiempo de ejecución.
11. Para probar el archivo SWF, seleccione Control > Probar película.

    En el componente TextInput del escenario, introduzca un código no válido de Estados Unidos, por ejemplo, uno que contenga sólo letras o uno que contenga menos de cinco números. Observe los mensajes de error que aparecen en el panel Salida.

Clase DataType (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.DataType

La clase DataType ofrece un acceso de lectura y escritura a los campos de datos de una propiedad de componente. Para obtener un objeto DataType, llame al método `ComponentMixins.getField()` de un componente. A continuación, puede llamar a métodos del objeto DataType para obtener y definir el valor del campo.

Si obtiene y establece valores de campos directamente en la instancia de componente en lugar de usar métodos de la clase DataType, se proporcionarán datos "sin formato". En cambio, si se obtienen o establecen valores de campos mediante métodos de DataType, los valores se procesan según la configuración de esquema del campo.

Por ejemplo, mediante el código siguiente se obtiene el valor de una propiedad de componente directamente, que se asigna a una variable. La variable, `propVar`, contiene el valor actual sin formato de la propiedad `propName`.

```
var propVar = myComponent.propName;
```

En el ejemplo siguiente se obtiene el valor de la misma propiedad mediante el método `DataType.getAsString()`. En este caso, el valor asignado a `stringVar` es el valor de `propName` una vez que se haya procesado según la configuración de su esquema y, a continuación, se devuelve en forma de cadena.

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var stringVar: String = dataTypeObj.getAsString();
```

Para más información sobre cómo especificar la configuración de esquema de un campo, vea "Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)" en *Utilización en Flash*.

También puede utilizar los métodos de la clase DataType para obtener o definir campos en diferentes tipos de datos. La clase DataType convierte automáticamente los datos sin formato al tipo deseado, si es posible. Por ejemplo, en el ejemplo anterior de códigos, los datos recuperados se convierten en el tipo String, aunque los datos sin formato sean de otro tipo.

El método `ComponentMixins.getField()` se encuentra disponible para los componentes que se han incluido en una vinculación de datos (ya sea como origen, destino o índice) o que se han inicializado mediante el método `ComponentMixins.initComponent()`. Para más información, consulte "Clase ComponentMixins (sólo en Flash Professional)" en la página 237.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de métodos de la clase DataType

En la tabla siguiente se enumeran los métodos de la clase DataType.

Método	Descripción
`DataType.getAnyTypedValue()`	Recoge el valor actual del campo.
`DataType.getAsBoolean()`	Recoge el valor actual del campo como valor booleano.
`DataType.getAsNumber()`	Recoge el valor actual del campo como un número.
`DataType.getAsString()`	Recoge el valor actual del campo como valor de cadena.
`DataType.getTypedValue()`	Recoge el valor actual del campo con el formato del tipo de datos solicitado.
`DataType.setAnyTypedValue()`	Establece un nuevo valor en el campo.
`DataType.setAsBoolean()`	Establece el campo según el nuevo valor, que se proporciona en forma de valor booleano.
`DataType.setAsNumber()`	Establece el campo según el nuevo valor, que se proporciona en forma de número.
`DataType.setAsString()`	Establece el campo según el nuevo valor, que se proporciona en forma de cadena.
`DataType.setTypedValue()`	Establece un nuevo valor en el campo.

Resumen de propiedades de la clase DataType

En la tabla siguiente se enumeran las propiedades de la clase DataType.

Propiedad	Descripción
`DataType.encoder`	Proporciona una referencia para el objeto Encoder asociado con este campo.
`DataType.formatter`	Proporciona una referencia para el objeto Formatter asociado con este campo.
`DataType.kind`	Proporciona una referencia para el objeto Kind asociado con este campo.

DataType.encoder

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.encoder

Descripción

Propiedad; proporciona una referencia para el objeto Encoder asociado con este campo, en caso de que exista. Puede utilizar esta propiedad para acceder a todas las propiedades y los métodos definidos mediante el codificador específico aplicado al campo de la ficha Esquema del inspector de componentes.

Si no se ha aplicado ningún codificador al campo en cuestión, esta propiedad devolverá el valor `undefined`.

Para más información sobre los codificadores proporcionados con Flash, consulte "Codificadores de esquema" en *Utilización de Flash*.

Ejemplo

En el ejemplo siguiente se supone que el campo al que se accede (`isValid`) utiliza un codificador Boolean (`mx.data.encoders.Bool`). Este codificador se proporciona con Flash y contiene una propiedad denominada `trueStrings` que especifica las cadenas que deben interpretarse como valores `true`. El código siguiente establece que la propiedad `trueStrings` del codificador de un campo sean las cadenas "Yes" y "Oui".

```
var myField:mx.data.binding.DataType = dataSet.getField("isValid");
myField.encoder.trueStrings = "Yes,Oui";
```

DataType.formatter

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.formatter

Descripción

Propiedad; proporciona una referencia para el objeto formateador asociado con este campo, en caso de que exista. Puede utilizar esta propiedad para acceder a todas las propiedades y métodos del objeto Formatter que se aplica al campo en la ficha Esquema del inspector de componentes.

Si no se ha aplicado ningún formateador al campo en cuestión, esta propiedad devolverá el valor `undefined`.

Para más información sobre los formateadores proporcionados con Flash, consulte "Formateadores de esquema" en *Utilización de Flash*.

Ejemplo

En este ejemplo se supone que el campo al que se accede está utilizando el formateador de número (`mx.data.formatters.NumberFormatter`) que se proporciona con Flash Professional 8. Este formateador contiene una propiedad denominada `precision` que especifica el número de dígitos que deben visualizarse después de una coma decimal. Este código establece la propiedad `precision` en dos cifras decimales para el campo que utiliza dicho formateador.

```
var myField:DataType = dataGrid.getField("currentBalance");
myField.formatter.precision = 2;
```

DataType.getAnyTypedValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.getAnyTypedValue(*suggestedTypes*)

Parámetros

suggestedTypes Matriz de cadenas que especifica, en el orden descendente que desee, los tipos de datos preferidos para el campo.

Valor devuelto

Valor actual del campo, con el formato de uno de los tipos de datos especificados en la matriz *suggestedTypes*.

Descripción

Método; recoge el valor actual del campo, utilizando la información del esquema del campo, para procesar el valor. Si el campo puede proporcionar un valor como primer tipo de datos especificado en la matriz *suggestedTypes*, el método devuelve el valor del campo con ese tipo de datos. De lo contrario, el método intentará extraer el valor del campo como segundo tipo de datos especificado en la matriz *suggestedTypes*, etc.

Si especifica `null` como uno de los elementos de la matriz *suggestedTypes*, el método devolverá el valor del campo del tipo de datos especificado en el panel Esquema del inspector de componentes. Si especifica `null` siempre se devolverá un valor; por lo tanto, utilice `null` sólo al final de la matriz.

Si un valor no puede devolverse con el formato de uno de los tipos sugeridos, se devolverá en el tipo especificado en la ficha Esquema.

Ejemplo

En este ejemplo se intenta obtener el valor de un campo (`productInfo.available`) en la propiedad `results` de un componente XMLConnector, primero como un número y, si no funciona, como una cadena.

```
import mx.data.binding.DataType;
import mx.data.binding.TypedValue;
var f: DataType = myXmlConnector.getField("results",
  "productInfo.available");
var b: TypedValue = f.getAnyTypedValue(["Number", "String"]);
```

Véase también

`ComponentMixins.getField()`

DataType.getAsBoolean()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.getAsBoolean()

Parámetros

Ninguno.

Valor devuelto

Un valor booleano.

Descripción

Método; recoge el valor actual del campo y lo convierte a forma booleana si es necesario.

Ejemplo

En este ejemplo, el campo denominado `propName` que pertenece a un componente denominado `myComponent` se recupera como valor booleano y se asigna a una variable:

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:Boolean = dataTypeObj.getAsBoolean();
```

DataType.getAsNumber()

Disponibilidad

Flash Player 6.

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.getAsNumber()

Parámetros

Ninguno.

Valor devuelto

Un número.

Descripción

Método; recoge el valor actual del campo y lo convierte a forma numérica si es necesario.

Ejemplo

En este ejemplo, el campo denominado `propName` que pertenece a un componente denominado `myComponent` se recupera en forma de número y se asigna a una variable:

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:Number = dataTypeObj.getAsNumber();
```

Véase también

`DataType.getAnyTypedValue()`

DataType.getAsString()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`dataTypeObject``.getAsString()`

Parámetros

Ninguno.

Valor devuelto

Una cadena.

Descripción

Método; recoge el valor actual del campo y lo convierte en una cadena si es necesario.

Ejemplo

En este ejemplo, una propiedad denominada `propName` que pertenece a un componente denominado `myComponent` se recupera en forma de cadena y se asigna a una variable:

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:String = dataTypeObj.getAsString();
```

Véase también

`DataType.getAnyTypedValue()`

DataType.getTypedValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.getTypedValue(*requestedType*)

Parámetros

requestedType Cadena que contiene el nombre de un tipo de datos o `null`.

Valor devuelto

Un objeto TypedValue (véase "Clase TypedValue (sólo en Flash Professional)" en la página 256).

Descripción

Método; devuelve el valor del campo en el formulario especificado, si el campo puede proporcionar su valor en ese formulario. Si el campo no puede proporcionar su valor en el formulario solicitado, el método devuelve `null`.

Si se especifica `null` como *requestedType*, el método devolverá el valor del campo en su tipo predeterminado.

Ejemplo

El ejemplo siguiente devuelve el valor del campo convertido al tipo de datos Boolean. Este valor se almacena en la variable `bool`.

```
var bool:TypedValue = field.getTypedValue("Boolean");
```

DataType.kind

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.kind

Descripción

Propiedad; proporciona una referencia al objeto Kind asociado con este campo. Puede utilizar esta propiedad para acceder a las propiedades y los métodos del objeto Kind.

DataType.setAnyTypedValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.setAnyTypedValue(*newTypedValue*)

Parámetros

newTypedValue Valor del objeto TypedValue que debe definirse en el campo. Para más información, consulte "Clase TypedValue (sólo en Flash Professional)" en la página 256.

Valor devuelto

Matriz de cadenas que describen los errores que se producen mientras se intenta definir un valor nuevo. Los errores pueden producirse en una de las condiciones siguientes:

- Los datos proporcionados no pueden convertirse en el tipo de datos de este campo (por ejemplo, "abc" no puede convertirse en un tipo de datos Number).
- El tipo de los datos es correcto pero no cumple con los criterios de validación del campo.
- El campo es de sólo lectura.

NOTA	El texto de un mensaje de error varía en función del tipo de datos, los formateadores y los codificadores definidos en el esquema del campo.

Descripción

Método; define un valor nuevo en el campo, utilizando la información del esquema del campo para procesar el campo.

Este método funciona llamando primero a DataType.setTypedValue() para establecer el valor. Si no funciona, el método comprueba si el objeto de destino aceptará datos String, Boolean o Number; en tal caso, los intentos de utilizar la conversión en ActionScript funcionarán correctamente.

Ejemplo

En este ejemplo se crea un nuevo objeto TypedValue (valor booleano) y, a continuación, se asigna dicho valor al objeto DataType denominado `field`. Todos los errores que se produzcan se asignan a la matriz `errors`.

```
import mx.data.binding.*;
var t:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setAnyTypedValue (t);
```

Véase también

`DataType.setTypedValue()`

DataType.setAsBoolean()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`dataTypeObject``.setAsBoolean(`*`newBooleanValue`*`)`

Parámetros

`newBooleanValue` Valor booleano.

Valor devuelto

Ninguno.

Descripción

Método; establece el campo según el valor nuevo, que se proporciona como un valor booleano. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

Ejemplo

En el ejemplo siguiente se establece una variable denominada `bool` en el valor booleano `true`. A continuación, se establece el valor al que hace referencia `field` en `true`.

```
var bool: Boolean = true;
field.setAsBoolean (bool);
```

DataType.setAsNumber()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.setAsNumber(*newNumberValue*)

Parámetros

newNumberValue Número.

Valor devuelto

Ninguno.

Descripción

Método; establece el campo según el valor nuevo, que se proporciona en forma de número. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

Ejemplo

En el ejemplo siguiente se establece una variable denominada `num` en el valor numérico 32. A continuación, se establece el valor al que hace referencia `field` en `num`.

```
var num: Number = 32;
field.setAsNumber (num);
```

DataType.setAsString()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

dataTypeObject.setAsString(*newStringValue*)

Parámetros

newStringValue Cadena.

Valor devuelto

Ninguno.

Descripción

Método; establece el campo según el valor nuevo, que se proporciona en forma de cadena. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

Ejemplo

En el ejemplo siguiente se establece la variable `stringVal` en la cadena `"The new value"`. A continuación, se establece el valor `field` en la cadena.

```
var stringVal: String = "The new value";
field.setAsString (stringVal);
```

DataType.setTypedValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`dataTypeObject``.setTypedValue(`*`newTypedValue`*`)`

Parámetros

`newTypedValue` Valor del objeto TypedValue que debe definirse en el campo.

Para más información acerca de los objetos TypedValue, consulte "Clase TypedValue (sólo en Flash Professional)" en la página 256.

Valor devuelto

Matriz de cadenas que describen los errores que se producen mientras se intenta definir un valor nuevo. Los errores pueden producirse en una de las condiciones siguientes:

- El tipo de datos que se proporciona no es aceptable.
- Los datos proporcionados no pueden convertirse en el tipo de datos de este campo (por ejemplo, `"abc"` no puede convertirse en un tipo de datos Number).
- El tipo de los datos es correcto pero no cumple con los criterios de validación del campo.
- El campo es de sólo lectura.

NOTA	El texto de un mensaje de error varía en función del tipo de datos, los formateadores y los codificadores definidos en el esquema del campo.

Descripción

Método; define un valor nuevo en el campo, utilizando la información del esquema del campo para procesar el campo. Este método se comporta de forma similar a `DataType.setAnyTypedValue()`, salvo en que no realiza tantos intentos de convertir los datos en un tipo de datos aceptable. Para más información, consulte `DataType.setAnyTypedValue()`.

Ejemplo

En este ejemplo se crea un nuevo objeto TypedValue (valor booleano) y, a continuación, se asigna dicho valor al objeto DataType denominado `field`. Todos los errores que se produzcan se asignan a la matriz `errors`.

```
import mx.data.binding.*;
var bool:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setTypedValue (bool);
```

Véase también

`DataType.setTypedValue()`

Clase TypedValue (sólo en Flash Professional)

Nombre de clase de ActionScript mx.data.binding.TypedValue

Un objeto TypedValue contiene un valor de datos e información sobre el tipo de datos del valor. Varios métodos de la clase DataType devuelven objetos TypedValue. Estos objetos se proporcionan en forma de parámetros para dichos métodos. La información de tipo de datos del objeto TypedValue resulta útil para que los objetos DataType decidan el momento y el modo en los que se debe llevar a cabo la conversión de tipo.

NOTA	Para que esta clase esté disponible en tiempo de ejecución, debe incluir clases de vinculación de datos en el archivo FLA.

Para ver información general sobre las clases del paquete mx.data.binding, consulte "Clases del paquete mx.data.binding (sólo en Flash Professional)" en la página 218.

Resumen de propiedades de la clase TypedValue

En la tabla siguiente se enumeran las propiedades de la clase TypedValue.

Propiedad	Descripción
`TypedValue.type`	Contiene el esquema asociado con el valor del objeto TypedValue.
`TypedValue.typeName`	Asigna un nombre al tipo de datos del valor del objeto TypedValue.
`TypedValue.value`	Contiene el valor de datos del objeto TypedValue.

Constructor para la clase TypedValue

Disponibilidad

Flash Player 6 (6.0.79.0).

Sintaxis

```
new mx.data.binding.TypedValue(value, typeName, [type])
```

Parámetros

value Un valor de datos de cualquier tipo.

typeName Una cadena que contiene el nombre del tipo de datos del valor.

type Objeto opcional del esquema que describe el esquema de los datos más detalladamente. Este campo sólo es obligatorio en determinadas circunstancias, como por ejemplo, al establecer los datos en la propiedad `dataProvider` de un componente DataSet.

Descripción

Constructor; crea un nuevo objeto TypedValue.

TypedValue.type

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

typedValueObject.type

Descripción

Propiedad; contiene el esquema asociado con el valor del objeto TypedValue.

Ejemplo

Este ejemplo muestra `null` en el panel Salida:

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.type);
```

TypedValue.typeName

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

typedValueObject.typeName

Descripción

Propiedad; contiene el nombre del tipo de datos del valor del objeto TypedValue.

Ejemplo

Este ejemplo muestra `Boolean` en el panel Salida:

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.typeName);
```

TypedValue.value

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

typedValueObject.value

Descripción

Propiedad; contiene el valor de datos del objeto TypedValue.

Ejemplo

Este ejemplo muestra `true` en el panel Salida:

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.value);
```

CAPÍTULO 10

Componente DataGrid (sólo en Flash Professional)

10

El componente DataGrid permite crear visualizaciones y aplicaciones de datos con muchas posibilidades. Puede utilizar el componente DataGrid para crear instancias de un juego de registros (recuperado de una consulta de base de datos de Macromedia ColdFusion, Java o .Net) mediante Macromedia Flash Remoting y mostrarlo en columnas. También puede utilizar datos de un juego de datos o de una matriz para definir un componente DataGrid. La versión 2 del componente DataGrid se ha mejorado para incluir desplazamiento horizontal, mejor compatibilidad con eventos (incluida la compatibilidad con eventos para celdas editables), capacidades de clasificación mejoradas y mejoras de rendimiento.

Puede cambiar el tamaño y personalizar características como la fuente, el color y los bordes de columna de una cuadrícula. Puede utilizar un clip de película personalizado como procesador de celdas para cualquier columna de una cuadrícula. Los procesadores de celdas muestran el contenido de una celda. Puede utilizar las barras de desplazamiento para mover los datos de una cuadrícula; también puede desactivar las barras de desplazamiento y utilizar los métodos DataGrid para crear una visualización de estilo de vista de página. Para más información sobre la personalización, consulte "Clase DataGridColumn (sólo en Flash Professional)" en la página 315.

Si añade el componente DataGrid a una aplicación, puede utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al componente. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad del componente DataGrid:

```
mx.accessibility.DataGridAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Interacción con el componente DataGrid (sólo en Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente DataGrid.

Si el valor de `DataGrid.sortableColumns` y `DataGridColumn.sortOnHeaderRelease` es `true`, al hacer clic en un encabezado de columna la cuadrícula se ordenará según los valores de las celdas de la columna.

Si el valor de `DataGrid.resizableColumns` es `true`, al hacer clic en el área entre las dos columnas se podrá cambiar el tamaño de las mismas.

Si hace clic en una celda editable, ésta quedará seleccionada; si hace clic en una celda no editable, no quedará seleccionada. Una celda individual puede editarse cuando las propiedades `DataGrid.editable` y `DataGridColumn.editable` de la celda sean `true`.

Si se selecciona una instancia DataGrid haciendo clic en la misma o utilizando la tabulación, podrá utilizar las teclas siguientes para controlar dicha instancia:

Tecla	Descripción
Flecha abajo	Cuando se edita una celda, el punto de inserción se desplaza al final del texto de la celda. Si una celda no puede editarse, la tecla de flecha abajo permite gestionar la selección tal como lo hace el componente List.
Flecha arriba	Cuando se edita una celda, el punto de inserción se desplaza al principio del texto de la celda. Si una celda no puede editarse, la tecla de flecha arriba permite gestionar la selección tal como lo hace el componente List.
Flecha derecha	Cuando se edita una celda, el punto de inserción se desplaza un carácter hacia la derecha. Si una celda no puede editarse, la tecla de flecha derecha no hará nada.
Flecha izquierda	Cuando se edita una celda, el punto de inserción se desplaza un carácter hacia la izquierda. Si una celda no puede editarse, la tecla de flecha izquierda no hará nada.
Retorno/Intro/ Mayús+Intro	Si una celda puede editarse, se lleva a cabo el cambio y el punto de inserción se desplaza a la celda de la misma columna, en la fila siguiente (arriba o abajo, en función del sentido de desplazamiento activado).
Mayús+Tabulador/ Tabulador	Desplaza la selección al elemento anterior. Si presiona la tecla Tabulador, la selección de la última columna de la cuadrícula pasa a la primera columna de la línea siguiente. Si se presiona Mayús+Tabulador, el ajuste se llevará a cabo al revés. Se seleccionará todo el texto de la celda seleccionada.

Utilización del componente DataGrid (sólo en Flash Professional)

Puede utilizar el componente DataGrid como base para los numerosos tipos de aplicaciones basadas en datos. Puede visualizar fácilmente una vista de tabla con formato de una consulta de base de datos (o de otros datos). Sin embargo, también puede utilizar las funciones del procesador de celdas para crear partes más sofisticadas y editables de la interfaz de usuario. A continuación se presentan usos prácticos del componente DataGrid:

- Cliente de correo Web
- Páginas de resultados de búsqueda
- Aplicaciones de hojas de cálculo, como las calculadoras de préstamos y las aplicaciones de impresos de declaración de la renta.

Aspectos básicos de diseño del componente DataGrid

El componente DataGrid amplía el Componente List. Al diseñar una aplicación con el componente DataGrid, resulta útil comprender cómo está diseñada la clase List subyacente. A continuación, se detallan algunos requisitos y suposiciones fundamentales que Macromedia ha usado para desarrollar la clase List:

- Cuanto más pequeño, rápido y simple, mejor.

  Es preferible no hacer algo más complicado de lo que sea absolutamente necesario. Esta ha sido la directiva de diseño más importante; la mayoría de los requisitos que se enumeran a continuación se basan en esta directiva.
- Las listas tienen alturas de fila uniformes.

  Cada fila debe tener la misma altura; ésta se puede definir durante la edición o en tiempo de ejecución.
- La escala de las listas debe ajustarse a miles de registros.
- Las listas no miden el texto.

  Esto crea un problema de desplazamiento horizontal para componentes List y Tree; para obtener más información, consulte “Introducción al diseño del componente List” en la página 790. Sin embargo, el componente DataGrid admite `"auto"` como valor de `hScrollPolicy`, ya que mide columnas (que tienen la misma anchura por elemento), no texto.

El hecho de que las listas no midan texto explica por qué las listas tienen alturas de fila uniformes. Cambiar el tamaño de cada fila para que se ajuste al texto precisa demasiadas mediciones. Por ejemplo, si desea mostrar de forma precisa las barras de desplazamiento de una lista con altura de fila no uniforme, no es necesario medir previamente cada fila.

- Las listas funcionan peor como función de sus filas visibles.

  Aunque hay listas que pueden mostrar 5000 registros, no pueden representar 5000 registros a la vez. Cuantas más filas visibles (especificadas por la propiedad `rowCount`) tenga en el escenario, más trabajo deberá llevar a cabo la lista para generar sus representaciones. Limitar el número de filas visibles, si es posible, supone la mejor solución.

- Las listas no son tablas.

  Los componentes DataGrid proporcionan una interfaz para muchos registros. No están diseñados para mostrar toda la información, sino para mostrar la suficiente información como para que los usuarios puedan ampliarla posteriormente. La vista de mensajes en Microsoft Outlook es un excelente ejemplo. No se lee todo el mensaje de correo electrónico en la cuadrícula; resultaría difícil leerlo y el rendimiento del cliente sería bajo. Outlook muestra suficiente información para que el usuario vaya al elemento expuesto y vea más detalles.

Aspectos básicos del componente DataGrid: vista y modelo de datos

Conceptualmente, el componente DataGrid está compuesto por un modelo de datos y una vista que muestra los datos. El modelo de datos consta de tres partes principales:

- Proveedor de datos (DataProvider)

  Lista de elementos con los que se rellena una cuadrícula de datos. A las matrices que se encuentran en el mismo fotograma en forma de componente DataGrid se les asignan métodos automáticamente (desde la API de DataProvider) que permiten manipular datos y difundir cambios en varias vistas. Todos los objetos que implementen la API DataProvider pueden asignarse a la propiedad `DataGrid.dataProvider`, incluidos los juegos de registros, los juegos de datos, etc. El código siguiente crea un proveedor de datos llamado `myDP`:

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```

- Elemento

  Objeto de ActionScript que sirve para almacenar las unidades de información en las celdas de una columna. En realidad, una cuadrícula de datos es una lista que puede mostrar más de una columna de datos. Una lista puede concebirse como una matriz; cada espacio indexado de la lista es un elemento. En el caso del componente DataGrid, cada elemento está formado por campos. En el código siguiente, el contenido que se encuentra entre llaves ({}) es un elemento:

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```

- Campo

  Los campos son identificadores que indican los nombres de las columnas de los elementos. Esto corresponde a la propiedad `columnNames` de la lista de columnas. En el componente List, los campos suelen ser `label` y `data`, pero en el caso del componente DataGrid los campos pueden ser cualquier identificador. En el código siguiente, los campos son `name` y `price`:

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```

La vista está formada por tres partes principales:

- Fila

  Se trata de un objeto de la vista que se ocupa de presentar los elementos de la cuadrícula estableciendo celdas. Cada fila está dispuesta de forma horizontal debajo de la anterior.

- Columna

  Las columnas son campos que se muestran en la cuadrícula; cada campo corresponde a la propiedad `columnName` de cada columna.

  Cada columna es un objeto de vista (una instancia de la clase DataGridColumn) que se encarga de mostrar cada columna (por ejemplo, la anchura, el color, el tamaño, etc.).

  Existen tres formas de añadir columnas a una cuadrícula de datos: asignar un objeto DataProvider a `DataGrid.dataProvider` (de este modo se genera automáticamente una columna para cada campo del primer elemento), establecer `DataGrid.columnNames` para especificar los campos que se mostrarán o utilizar el constructor de la clase DataGridColumn para crear columnas y llamar a `DataGrid.addColumn()` para añadirlas a la cuadrícula.

  Para aplicar formato a las columnas, configure las propiedades de estilo para toda la cuadrícula de datos o defina los objetos DataGridColumn, configure los formatos de estilo de los mismos de forma individual y añádalos a la cuadrícula de datos.

- Celda

  Se trata de un objeto de la vista que se encarga de presentar los campos individuales para cada elemento. Para comunicarse con la cuadrícula de datos, estos componentes deben implementar la API CellRenderer (véase "Interfaz API CellRenderer" en la página 113). En el caso de las cuadrículas de datos básicas, las celdas son objetos TextField de ActionScript incorporados.

Parámetros de DataGrid

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DataGrid en el inspector de propiedades o en el inspector de componentes:

editable es un valor booleano que indica si la cuadrícula es editable (`true`) o no (`false`). El valor predeterminado es `false`.

multipleSelection es un valor booleano que indica si pueden seleccionarse varios elementos (`true`) o no (`false`). El valor predeterminado es `false`.

rowHeight indica la altura de cada una de las filas, en píxeles. Al modificar el tamaño de fuente no se altera la altura de la fila. El valor predeterminado es 20.

Es posible escribir ActionScript para controlar éstas y otras opciones adicionales para el componente DataGrid mediante sus propiedades, métodos y eventos. Para más información, consulte "Clase DataGrid (sólo en Flash Professional)" en la página 275.

Creación de aplicaciones con el componente DataGrid

Para crear una aplicación con el componente DataGrid, primero debe determinar el origen de los datos. Los datos de una cuadrícula provienen de un juego de registros basado en una consulta de base de datos de Macromedia ColdFusion, Java o .Net mediante Flash Remoting. Los datos también pueden provenir de un juego de datos o de una matriz. Para colocar los datos en una cuadrícula, establezca la propiedad `DataGrid.dataProvider` en el juego de registros, el juego de datos o la matriz. También puede utilizar los métodos de las clases DataGrid y DataGridColumn para crear datos de forma local. Un objeto Array del mismo fotograma que el componente DataGrid copia los métodos, las propiedades y los eventos de la API DataProvider.

NOTA	Cuando se vinculan datos al componente DataGrid mediante componentes Data, el objeto vincula columnas hacia atrás (similar a la reproducción indefinida de un objeto o matriz). Por lo tanto, para ordenar los datos en el componente DataGrid de forma diferente, es necesario definir de forma explícita las columnas.

Para utilizar Flash Remoting para añadir un componente DataGrid a una aplicación:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. En el panel Componentes, haga doble clic en el componente DataGrid para añadirlo al escenario.
3. En el inspector de propiedades, introduzca el nombre de instancia **myDataGrid**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

   ```
   myDataGrid.dataProvider = recordSetInstance;
   ```

   El juego de registros `recordSetInstance` de Flash Remoting se asigna a la propiedad `dataProvider` de `myDataGrid`.
5. Seleccione Control > Probar película.

Para utilizar un proveedor de datos local para añadir un componente DataGrid a una aplicación:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. En el panel Componentes, haga doble clic en el componente DataGrid para añadirlo al escenario.
3. En el inspector de propiedades, introduzca el nombre de instancia **myDataGrid**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

   ```
   myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
     price:"Cheap"});
   myDataGrid.dataProvider = myDP;
   ```

   Los campos `name` y `price` se utilizan como encabezados de columna y sus valores sirven para rellenar las celdas de cada fila.
5. Seleccione Control > Probar película.

Para especificar columnas y añadir la clasificación de un componente DataGrid en una aplicación:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. En el panel Componentes, haga doble clic en el componente DataGrid para añadirlo al escenario.
3. En el inspector de propiedades, introduzca el nombre de instancia **myDataGrid**.

4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
var myDataGrid:mx.controls.DataGrid;

// Crear columnas para permitir la ordenación de datos.
myDataGrid.addColumn("name");
myDataGrid.addColumn("score");

var myDP_array:Array = new Array({name:"Clark", score:3135},
  {name:"Bruce", score:403}, {name:"Peter", score:25})

myDataGrid.dataProvider = myDP_array;

// Crear un objeto detector para DataGrid.
var listener_obj:Object = new Object();
listener_obj.headerRelease = function(evt_obj:Object) {
 switch (evt_obj.target.columns[evt_obj.columnIndex].columnName) {
 case "name" :
  myDP_array.sortOn("name", Array.CASEINSENSITIVE);
  break;
 case "score" :
  myDP_array.sortOn("score", Array.NUMERIC);
  break;
 }
};

// Añadir un detector a DataGrid.
myDataGrid.addEventListener("headerRelease", listener_obj);
```

5. Seleccione Control > Probar película.

Para crear una instancia del componente DataGrid mediante código ActionScript:

1. Arrastre el componente DataGrid desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
this.createClassObject(mx.controls.DataGrid, "my_dg", 10,
  {columnNames:["name", "score"]});
my_dg.setSize(140, 100);
my_dg.move(10, 40);
```

   Este script utiliza el método `UIObject.createClassObject()` para crear la instancia de DataGrid y, a continuación, cambia de tamaño y coloca la cuadrícula.

3. Cree una matriz, añádale datos e identifique la matriz como el proveedor de datos para la cuadrícula de datos:

```
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;
```

4. Seleccione Control > Probar película.

Estrategias de rendimiento de DataGrid

El rendimiento puede convertirse rápidamente en una cuestión importante al utilizar el componente DataGrid, ya que puede cambiar el tamaño de los datos que muestra. Una cuadrícula de datos que muestre cien filas en un equipo rápido con una conexión rápida al origen de datos puede ser aceptable para el usuario. Pero un mes más tarde, cuando los datos han aumentado a varios miles de filas, es posible que el usuario no esté igual de satisfecho. Además, el usuario puede tener un equipo más lento y una conexión lenta al origen de datos.

A continuación se ofrecen algunas sugerencias para evitar problemas comunes de rendimiento cuando se utilizan componentes DataGrid.

- Cree y vincule una estructura de datos en lugar de añadir las columnas directamente.

  Existen dos formas de añadir columnas y datos al componente DataGrid: vincular estructuras de datos creadas previamente (una matriz de objetos) mediante la propiedad `DataGrid.dataProvider` o utilizar métodos de la clase DataGrid como `DataGrid.addColumn()` y `DataGrid.addItem()`. Siempre que sea posible, se debe vincular una estructura de datos creada previamente con la propiedad `DataGrid.dataProvider` porque permite al componente DataGrid crear todas las columnas que necesita antes de intentar dibujarlas en la pantalla.

  Es posible que tenga la tentación de crear un bucle `for` para llamar a `DataGrid.addColumn()` en todas las columnas necesarias. Aunque pueda parecer un enfoque simple y obvio, debe evitarse. Cada vez que se llama a `DataGrid.addColumn()`, la cuadrícula de datos añade detectores de eventos, se ordena y se vuelve a dibujar para presentar la columna nueva. Si se crean 20 columnas con `DataGrid.addColumn()`, el componente DataGrid se ordena y se vuelve a dibujar 20 veces sin necesidad. Si se vincula la estructura de datos en ActionScript, no es necesario tener en cuenta ninguna representación ni evento. Cuando se asigna a la propiedad `dataProvider` del componente DataGrid, se completa todo el dibujo en una sola pasada.

- Proporcione un mecanismo de ampliación para datos detallados.

  La interfaz del componente DataGrid permite a los usuarios realizar búsquedas más rápidas y detalladas. Proporcione únicamente los datos necesarios para realizar la búsqueda inicial y reserve la información detallada de una determinada fila o columna para el segundo paso de la búsqueda. Este proceso minimiza tanto los datos iniciales necesarios para completar la cuadrícula de datos como la cantidad de información que los usuarios tienen que leer para encontrar lo que buscan. Una vez que se ha seleccionado una fila o elemento de interés en la cuadrícula de datos, se puede realizar una segunda llamada al origen de datos para obtener información relacionada. Esta información se mostrará mejor en algún otro mecanismo de la IU como pueda ser una colección de gráficos y campos de texto de varias líneas.
- Evite la manipulación de ciclos de datos entre el origen de datos y la cuadrícula de datos.

  Si es posible y se siguen satisfaciendo las necesidades de la base de datos a largo plazo, almacenar los datos en el mismo formato y orden en el que se van a visualizar puede evitar aumentar innecesariamente la asignación de memoria y el tiempo de procesamiento en el equipo del usuario, así como acelerar el tiempo de respuesta de la cuadrícula de datos.
- Evite las consultas que devuelven todas las filas de la base de datos.

  Normalmente, un usuario no desea ver todos los registros cada vez que accede a los datos. Hay que saber lo que buscará el consumidor de los datos y ofrecerle los medios para que pueda acotar la búsqueda adecuadamente. Si normalmente se buscan sólo los registros más recientes de la semana para un tema determinado, muestre ese grupo menor de datos de forma predeterminada, de modo que se pueda ampliar la vista de los datos.

  Piense en la posibilidad de paginar grandes cantidades de datos para limitar su tamaño, mediante un subconjunto de datos que normalmente se devolvería de una consulta. Por ejemplo, en lugar de visualizar las 10.000 filas de datos que podrían devolverse de una consulta en la base de datos, se puede llamar a un subconjunto de las primeras 20 filas y unos botones de navegación adicionales pueden activar una llamada para rellenar la cuadrícula de datos con los siguientes 20 registros.

- Separe el procesamiento de datos del procesamiento de CellRenderer.

  La API CellRenderer permite mostrar contenido de celdas personalizado en una cuadrícula de datos. Un requisito funcional puede obligar al usuario a rellenar la cuadrícula de datos con un componente ComboBox u otro control de IU de forma condicional. Por ejemplo, en función de una selección de la columna dos, es posible volver a rellenar o seleccionar de forma automática opciones de la columna cuatro. Siempre que sea posible, es importante separar esta lógica condicional y volver a rellenar los controles del proceso de representación del contenido de la celda. Cada vez que el puntero del ratón pasa por encima de la celda, que ésta se selecciona o que cambian los datos, es probable que el contenido de la celda o que toda ella se vuelva a dibujar para mantenerla actualizada. Esto significa que cualquier código que incluya en CellRenderer se repetirá una y otra vez, por lo que debería mantener el procesamiento en CellRenderer lo más sencillo posible. Si necesita añadir código a CellRenderer, es mejor llamar a una función desde CellRenderer que detecte las actualizaciones necesarias y las lleve a cabo de la forma más eficaz.
- Utilice `UIObject.doLater()` para acceder a las propiedades cuando haya terminado de dibujarse la cuadrícula de datos.

  Para poder acceder a todas las propiedades de la cuadrícula de datos (por ejemplo, `focusedCell`, entre otras), es necesario que la instancia de cuadrícula de datos se termine de dibujar y cargue los datos. Dado que no hay ningún evento de "finalización" de un componente DataGrid, puede utilizar `UIObject.doLater()` en su lugar para llamar a una función que acceda a las propiedades de la cuadrícula de datos. `UIObject.doLater()` ejecutará la función únicamente cuando las propiedades de la cuadrícula de datos estén disponibles. Consulte `DataGrid.focusedCell` para ver un ejemplo.

Personalización del componente DataGrid (sólo en Flash Professional)

Puede transformar un componente DataGrid horizontal y verticalmente durante la edición y el tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`). Si no dispone de ninguna barra de desplazamiento horizontal, las anchuras de las columnas se ajustarán de forma proporcional. Si se ajusta el tamaño de una columna, y por lo tanto de una celda, es posible que el texto aparezca recortado.

Utilización de estilos con el componente DataGrid

Es posible definir propiedades de estilo para cambiar el aspecto de un componente DataGrid. El componente DataGrid hereda estilos del componente List. (Véase "Utilización de estilos con el componente List" en la página 794.) El componente DataGrid también es compatible con los estilos siguientes:

Estilo	Tema	Descripción
`backgroundColor`	Ambos	El color de fondo, que se puede establecer para toda la cuadrícula o para cada columna.
`backgroundDisabledColor`	Ambos	Color de fondo cuando el valor de la propiedad `enabled` del componente es `"false"`. El valor predeterminado es 0xDDDDDD (gris medio).
`borderStyle`	Ambos	El componente DataGrid utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es `"inset"`.
`headerColor`	Ambos	Color de los encabezados de columna. El valor predeterminado es 0xEAEAEA (gris claro)
`headerStyle`	Ambos	Declaración de estilo CSS para el encabezado de columna que se puede aplicar a una cuadrícula o columna con el fin de personalizar los estilos del encabezado.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).

`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. Por ejemplo (utilizando una instancia de DataGrid `my_dg`):
		`my_dg.setStyle("fontFamily", "yourFont");`
		`my_dg.embedFonts=true;`
		De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.

Estilo	Tema	Descripción
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`vGridLines`	Ambos	Valor booleano que indica si deben mostrarse las líneas verticales de la cuadrícula (`true`) o no (`false`). El valor predeterminado es `true`.
`hGridLines`	Ambos	Valor booleano que indica si deben mostrarse las líneas horizontales de la cuadrícula (`true`) o no (`false`). El valor predeterminado es `false`.
`vGridLineColor`	Ambos	Color de las líneas verticales de la cuadrícula. El valor predeterminado es 0x666666 (gris medio).
`hGridLineColor`	Ambos	Color de las líneas horizontales de la cuadrícula. El valor predeterminado es 0x666666 (gris medio).

Definición de estilos para una columna individual

Los estilos del color y el texto pueden establecerse para toda la cuadrícula o para una columna. Puede utilizar la sintaxis siguiente para establecer un estilo para una columna específica:

```
grid.getColumnAt(3).setStyle("backgroundColor", 0xFF00AA);
```

Definición de estilos de encabezado

Puede establecer estilos de encabezado mediante `headerStyle`, que es una propiedad de estilo. Para ello, debe crear una instancia de `CSSStyleDeclaration`, establecer las propiedades apropiadas en esa instancia para el encabezado y, a continuación, asignar el valor de `CSSStyleDeclaration` a la propiedad `headerStyle`, como se muestra en el siguiente ejemplo.

```
import mx.styles.CSSStyleDeclaration;
var headerStyles = new CSSStyleDeclaration();
headerStyles.setStyle("fontStyle", "italic");
grid.setStyle("headerStyle", headerStyles);
```

Definición de estilos de todos los componentes DataGrid de un documento

La clase DataGrid hereda de la clase List, que a su vez hereda de la clase ScrollSelectList. Las propiedades de estilo de clase predeterminadas se definen en la clase ScrollSelectList, que amplían el componente Menu y todos los componentes basados en List. Puede definir directamente en esta clase nuevos valores de estilo predeterminados y los nuevos valores se reflejarán en todos los componentes afectados.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Para definir una propiedad de estilo sólo en el componente DataGrid, puede crear una nueva instancia de `CSSStyleDeclaration` y almacenarla en `_global.styles.DataGrid`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.DataGrid == undefined) {
  _global.styles.DataGrid = new CSSStyleDeclaration();
}
_global.styles.DataGrid.setStyle("backgroundColor", 0xFF00AA);
```

Al crear una nueva declaración de estilo de nivel de clase, perderá todos los valores predeterminados proporcionados por la declaración de `ScrollSelectList`, incluido el de `backgroundColor`, que es necesario para los eventos del ratón. Para crear una declaración de estilo de clase y conservar los valores predeterminados, utilice un bucle `for..in` para copiar los valores anteriores a la nueva declaración.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.DataGrid;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Para más información sobre los estilos de clase, consulte “Definición de los estilos de una clase de componente” en *Utilización de componentes*.

Utilización de aspectos con el componente DataGrid

Los aspectos que el componente DataGrid utiliza para representar sus estados visuales se incluyen en los subcomponentes que constituyen la cuadrícula de datos (barras de desplazamiento y RectBorder). Para información sobre estos aspectos, véase "Utilización de aspectos con el componente UIScrollBar" en la página 1435 y "Clase RectBorder" en la página 1103.

Clase DataGrid (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > ScrollSelectList > Componente List > DataGrid

Nombre de clase de ActionScript mx.controls.DataGrid

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.DataGrid.version);
```

NOTA

El código `trace(myDataGridInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase DataGrid

En la tabla siguiente se enumeran los métodos de la clase DataGrid.

Método	Descripción
`DataGrid.addColumn()`	Añade una columna a la cuadrícula de datos.
`DataGrid.addColumnAt()`	Añade una columna a la cuadrícula de datos en una ubicación especificada.
`DataGrid.addItem()`	Añade un elemento a la cuadrícula de datos.
`DataGrid.addItemAt()`	Añade un elemento a la cuadrícula de datos en una ubicación especificada.
`DataGrid.editField()`	Reemplaza los datos de celda en una ubicación especificada.

Método	Descripción
`DataGrid.getColumnAt()`	Obtiene una referencia a una columna en una ubicación especificada.
`DataGrid.getColumnIndex()`	Obtiene una referencia al objeto DataGridColumn en el índice especificado.
`DataGrid.removeAllColumns()`	Elimina todas las columnas de una cuadrícula de datos.
`DataGrid.removeColumnAt()`	Elimina una columna de una cuadrícula de datos en una ubicación especificada.
`DataGrid.replaceItemAt()`	Reemplaza un elemento de una ubicación especificada por otro.
`DataGrid.spaceColumnsEqually()`	Distribuye espacios uniformes en todas las columnas.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase DataGrid de la clase UIObject. Al llamar a estos métodos, debe utilizarse la forma *`dataGridInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase DataGrid de la clase UIComponent. Al llamar a estos métodos, debe utilizarse la forma *dataGridInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Métodos heredados de la clase List

En la tabla siguiente se enumeran los métodos que hereda la clase DataGrid de la clase List. Al llamar a estos métodos, debe utilizarse la forma *dataGridInstance.methodName*.

Método	Descripción
List.addItem()	Añade un elemento al final de la lista.
List.addItemAt()	Añade un elemento a la lista en el índice especificado.
List.getItemAt()	Devuelve el elemento del índice especificado.
List.removeAll()	Elimina todos los elementos de la lista.
List.removeItemAt()	Elimina el elemento en el índice especificado.
List.replaceItemAt()	Sustituye por otro el elemento del índice especificado.
List.setPropertiesAt()	Aplica las propiedades especificadas al elemento especificado.
List.sortItems()	Ordena los elementos de la lista según la función de comparación especificada.
List.sortItemsBy()	Ordena los elementos de la lista según la propiedad especificada.

Resumen de propiedades de la clase DataGrid

En la tabla siguiente se enumeran las propiedades de la clase DataGrid.

Propiedad	Descripción
DataGrid.columnCount	Sólo lectura; el número de columnas que se muestran.
DataGrid.columnNames	Matriz de nombres de campo de cada elemento que aparecen en forma de columnas.
DataGrid.dataProvider	Modelo de datos de la cuadrícula de datos.

Propiedad	Descripción
DataGrid.editable	Valor booleano que indica si la cuadrícula de datos es editable (true) o no (false).
DataGrid.focusedCell	Define la celda que está seleccionada.
DataGrid.headerHeight	Altura de los encabezados de columna, expresada en píxeles.
DataGrid.hScrollPolicy	Indica si la barra de desplazamiento horizontal está activa ("on"), no lo está ("off") o aparece cuando es necesario ("auto").
DataGrid.resizableColumns	Valor booleano que indica si puede cambiarse el tamaño de las columnas (true) o no (false).
DataGrid.selectable	Valor booleano que indica si la cuadrícula de datos es seleccionable (true) o no (false).
DataGrid.showHeaders	Valor booleano que indica que los encabezados de columna están visibles (true) o no (false).
DataGrid.sortableColumns	Valor booleano que indica si las columnas pueden clasificarse (true) o no (false).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase DataGrid de la clase UIObject. Al acceder a estas propiedades desde el objeto DataGrid, debe utilizarse la forma *dataGridInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
UIObject.height	Altura del objeto, expresada en píxeles. Sólo lectura.
UIObject.left	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase DataGrid de la clase UIComponent. Al acceder a estas propiedades desde el objeto DataGrid, debe utilizarse la forma *dataGridInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase List

En la tabla siguiente se enumeran las propiedades que hereda la clase DataGrid de la clase List. Al acceder a estas propiedades desde el objeto DataGrid, debe utilizarse la forma *dataGridInstance.propertyName*.

Propiedad	Descripción
List.cellRenderer	Asigna la clase o el símbolo que se va utilizar para mostrar cada fila de la lista.
List.dataProvider	Origen de los elementos de la lista.
List.hPosition	Posición horizontal de la lista.

Propiedad	Descripción
`List.hScrollPolicy`	Indica si la barra de desplazamiento horizontal se muestra (`"on"`) o no (`"off"`).
`List.iconField`	Campo de cada elemento que se utiliza para especificar iconos.
`List.iconFunction`	Función que determina qué icono se debe utilizar.
`List.labelField`	Especifica el campo de cada elemento que se utilizará como texto de etiqueta.
`List.labelFunction`	Función que determina qué campos de cada elemento se utilizarán para el texto de etiqueta.
`List.length`	Número de elementos de la lista. Es una propiedad de sólo lectura.
`List.maxHPosition`	Número de píxeles que la lista puede desplazarse a la derecha cuando `List.hScrollPolicy` está definida en `"on"`.
`List.multipleSelection`	Indica si en la lista se permite la selección múltiple (`true`) o no (`false`).
`List.rowCount`	Número de filas que son al menos parcialmente visibles en la lista.
`List.rowHeight`	Altura de las filas de la lista, expresada en píxeles.
`List.selectable`	Indica si la lista es seleccionable (`true`) o no (`false`).
`List.selectedIndex`	Índice de una selección en una lista de selección única.
`List.selectedIndices`	Matriz de los elementos seleccionados en una lista de selección múltiple.
`List.selectedItem`	Elemento seleccionado en una lista de selección única. Es una propiedad de sólo lectura.
`List.selectedItems`	Objetos del elemento seleccionado en una lista de selección múltiple. Es una propiedad de sólo lectura.
`List.vPosition`	Desplaza la lista para que el elemento visible en la parte superior sea el número asignado.
`List.vScrollPolicy`	Indica si la barra de desplazamiento vertical se muestra (`"on"`), no se muestra (`"off"`) o se muestra cuando es necesario (`"auto"`).

Resumen de eventos de la clase DataGrid

En la tabla siguiente se enumeran los eventos de la clase DataGrid.

Evento	Descripción
`DataGrid.cellEdit`	Se difunde cuando se modifica el valor de celda.
`DataGrid.cellFocusIn`	Se difunde cuando se selecciona una celda.
`DataGrid.cellFocusOut`	Se difunde cuando se anula la selección de una celda.
`DataGrid.cellPress`	Se difunde cuando se presiona (se hace clic en) una celda.
`DataGrid.change`	Se difunde cuando se ha seleccionado un elemento.
`DataGrid.columnStretch`	Se difunde cuando un usuario cambia el tamaño de una columna horizontalmente.
`DataGrid.headerRelease`	Se difunde cuando un usuario hace clic (se suelta el botón del ratón) en un encabezado.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase DataGrid de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase DataGrid de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase List

En la tabla siguiente se enumeran los eventos que hereda la clase DataGrid de la clase List.

Evento	Descripción
`List.change`	Se difunde cuando la interacción del usuario provoca un cambio en la selección.
`List.itemRollOut`	Se difunde cuando el puntero del ratón se desplaza sobre elementos de la lista y después sale de ellos.
`List.itemRollOver`	Se difunde cuando el puntero del ratón se desplaza sobre elementos de la lista.
`List.scroll`	Se difunde cuando se recorre la lista.

DataGrid.addColumn()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDataGrid.addColumn(dataGridColumn)
myDataGrid.addColumn(name)
```

Parámetros

dataGridColumn Instancia de la clase DataGridColumn.

name Cadena que indica el nombre de un nuevo objeto DataGridColumn que debe insertarse.

Valor devuelto

Una referencia al objeto DataGridColumn que se ha añadido o la cadena que indica el nombre de la nueva columna.

Descripción

Método; añade una columna nueva al final de la cuadrícula de datos. Para más información, consulte "Clase DataGridColumn (sólo en Flash Professional)" en la página 315.

Ejemplo

Este ejemplo muestra tres formas diferentes de crear columnas para un componente DataGrid. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal (observe que importa primero la clase DataGridColumn):

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(320, 240);

// Añadir columnas a la cuadrícula.
my_dg.addColumn("Red");

// Añadir otra columna a la cuadrícula.
my_dg.addColumn(new DataGridColumn("Green"));

// Añadir una tercera columna a la cuadrícula.
var blue_dgc:DataGridColumn = new DataGridColumn("Blue");
blue_dgc.width = 140;
blue_dgc.headerText = "Blue Column:";
my_dg.addColumn(blue_dgc);
```

DataGrid.addColumnAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.addColumnAt(*index*, *name*)

myDataGrid.addColumnAt(*index*, *dataGridColumn*)

Parámetros

index Posición de índice en la que se añade el objeto DataGridColumn. La primera posición es 0.

name Cadena que indica el nombre del objeto DataGridColumn.

dataGridColumn Instancia de la clase DataGridColumn.

Valor devuelto

Una referencia al objeto DataGridColumn que se ha añadido o la cadena que indica el nombre de la nueva columna.

Descripción

Método; añade una columna nueva en la posición especificada. Las columnas se desplazan a la derecha y sus índices se incrementan. Para más información, consulte "Clase DataGridColumn (sólo en Flash Professional)" en la página 315.

Ejemplo

Este ejemplo muestra dos formas de utilizar `addColumnAt()` y define las anchuras de columna. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal (observe que importa primero la clase DataGridColumn):

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(320, 240);

// Añadir columnas a la cuadrícula.
my_dg.addColumnAt(0, "Orange");
var orange_dgc:DataGridColumn = my_dg.getColumnAt(0);
orange_dgc.width = 125;

var blue_dgc:DataGridColumn = new DataGridColumn("Blue");
blue_dgc.width = 75;
my_dg.addColumnAt(1, blue_dgc);
```

DataGrid.addItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.addItem(*item*)

Parámetros

item Instancia de un objeto que debe añadirse a una cuadrícula.

Valor devuelto

Referencia a la instancia que se ha añadido.

Descripción

Método; añade un elemento al final de la cuadrícula (después del último índice del elemento).

NOTA

Este método difiere del método `List.addItem()` en que se pasa un objeto en lugar de una cadena.

Ejemplo

Este ejemplo crea una columna con el encabezado "name" y, a continuación, inserta el valor `item_obj` para "name". Observe que el valor "age" se omite porque sólo se ha definido la columna name. Si no se especifica una columna (borre la línea `addColumn`), el componente DataGrid crea de forma automática las columnas apropiadas. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Añadir columnas a la cuadrícula y añadir datos.
my_dg.addColumn("name");

var item_obj:Object = {name:"Jim", age:30};
my_dg.addItem(item_obj);
```

DataGrid.addItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.addItemAt(*index*, *item*)

Parámetros

index Posición de índice (entre los nodos secundarios) en la que debe añadirse el nodo. La primera posición es 0.

item Cadena que muestra el nodo.

Valor devuelto

Referencia a la instancia de objeto que se ha añadido.

Descripción

Método; añade un elemento a la cuadrícula en la posición especificada.

Ejemplo

Este ejemplo crea una columna con el encabezado "name", rellena la columna desde una matriz y, a continuación, añade el nombre "Chase" en la primera fila. Observe que el valor "age" se omite porque sólo se ha definido la columna name. Si no se especifica una columna (borre la línea `addColumn`), el componente DataGrid crea de forma automática las columnas apropiadas. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
var my_dg:mx.controls.DataGrid;

// Añadir columnas a la cuadrícula y añadir datos.
my_dg.addColumn("name");

var myDP_array:Array = new Array({name:"John", age:33}, {name:"Jose",
  age:41});
my_dg.dataProvider = myDP_array;

var item_obj:Object = {name:"Chase", age:30};
my_dg.addItemAt(0, item_obj);
```

DataGrid.cellEdit

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.cellEdit = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("cellEdit", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se ha modificado un valor de celda.

Los componentes de la versión 2 de la arquitectura de componentes de Macromedia utilizan un modelo de evento distribuidor/detector. El componente DataGrid distribuye un evento `cellEdit` cuando se ha modificado el valor de una celda y el evento se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.cellEdit` tiene cuatro propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino.

`itemIndex` Número que indica el índice de la fila de destino.

`oldValue` Valor anterior de la celda.

`type` Cadena `"cellEdit"`.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `myDataGridListener` y se pasa al método `myDataGrid.addEventListener()` como segundo parámetro. El objeto de evento se captura mediante el controlador `cellEdit` del parámetro *`eventObject`*. Cuando se difunde el evento `cellEdit` (después de haber modificado un valor "score" y haber presionado Intro) se envía una sentencia `trace` al panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(320, 240);
my_dg.editable = true;

// Añadir columnas y hacer que la primera no sea editable.
my_dg.addColumn("name");
my_dg.getColumnAt(0).editable = false;
my_dg.addColumn("score");

var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

// Definir origen de datos de DataGrid.
my_dg.dataProvider = myDP_array;

// Crear un objeto detector.
var myListener_obj:Object = new Object();
myListener_obj.cellEdit = function(evt_obj:Object) {
 // Recuperar ubicación de celda modificada.
 var cell_obj:Object = "("+evt_obj.columnIndex+", "+evt_obj.itemIndex+")";
 // Recuperar valor de celda modificada.
 var value_obj:Object = evt_obj.target.selectedItem.score;
 trace("The value of the cell at "+cell_obj+" has changed to "+value_obj);
};

// Añadir objeto detector.
my_dg.addEventListener("cellEdit", myListener_obj);
```

DataGrid.cellFocusIn

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.cellFocusIn = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("cellFocusIn", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una celda determinada. Este evento se difunde cuando se difunden todos los eventos `editCell` y `cellFocusOut` de las celdas editadas anteriormente.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando un componente DataGrid distribuye un evento `cellFocusIn`, el evento se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.cellFocusIn` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino.

`itemIndex` Número que indica el índice de la fila de destino.

`type` Cadena `"cellFocusIn"`.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `dgListener` y se pasa al método `my_dg.addEventListener()` como segundo parámetro. Cuando se difunde el evento `cellFocusIn`, se envía una sentencia `trace` al panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

// Hacer que DataGrid sea editable.
my_dg.editable = true;

// Crear un objeto detector.
var dgListener:Object = new Object();
dgListener.cellFocusIn = function(evt_obj:Object) {
    var cell_str:String = "(" + evt_obj.columnIndex + ", " +
  evt_obj.itemIndex + ")";
    trace("The cell at " + cell_str + " has gained focus");
};

// Añadir detector.
my_dg.addEventListener("cellFocusIn", dgListener);
```

NOTA La cuadrícula debe ser editable para que este código funcione y el evento sólo se difunda para celdas editables. Por tanto, si tiene dos columnas y sólo una de ellas es editable (supongamos que "score"), entonces, al hacer clic en una fila en la columna "nombre" no se activará este evento.

DataGrid.cellFocusOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.cellFocusOut = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("cellFocusOut", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario sale de una celda que está seleccionada. Puede utilizar las propiedades del objeto de evento para aislar la celda de la que se ha salido. Este evento se difunde después del evento `cellEdit` y antes de que los eventos `cellFocusIn` posteriores se difundan para la celda siguiente.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando el componente DataGrid distribuye un evento `cellFocusOut`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.cellFocusOut` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino. La primera posición es 0.

`itemIndex` Número que indica el índice de la fila de destino. La primera posición es 0.

`type` La cadena `"cellFocusOut"`.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `dgListener` y se pasa al método `my_dg.addEventListener()` como segundo parámetro. Cuando se difunde el evento `cellFocusOut`, se envía una sentencia `trace` al panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;
```

```
// Hacer que DataGrid sea editable.
my_dg.editable = true;

// Crear un objeto detector.
var dgListener:Object = new Object();
dgListener.cellFocusOut = function(evt_obj:Object) {
    var cell_str:String = "(" + evt_obj.columnIndex + ", " +
  evt_obj.itemIndex + ")";
    trace("The cell at " + cell_str + " has lost focus");
};

// Añadir detector.
my_dg.addEventListener("cellFocusOut", dgListener);
```

NOTA	La cuadrícula debe ser editable para que este código funcione y el evento sólo se difunda para celdas editables. Por tanto, si tiene dos columnas y sólo una de ellas es editable (supongamos que "score"), al hacer clic en una fila en la columna "nombre" no se activará este evento.

DataGrid.cellPress

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.cellPress = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("cellPress", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario presiona el botón del ratón en una celda.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando el componente DataGrid difunde un evento `cellPress`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.cellPress` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna que se presionó. La primera posición es 0.

`itemIndex` Número que indica el índice de la fila que se presionó. La primera posición es 0.

`type` Cadena `"cellPress"`.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `dgListener` y se pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `cellPress` del parámetro *evt_obj*. Cuando se difunde el evento `cellPress`, se envía una sentencia `trace` al panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Configurar datos de ejemplo.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Crear un objeto detector.
var dgListener:Object = new Object();
dgListener.cellPress = function(evt_obj:Object) {
 var cell_str:String = "("+evt_obj.columnIndex+", "+evt_obj.itemIndex+")";
 trace("The cell at "+cell_str+" has been clicked");
};

// Añadir detector.
my_dg.addEventListener("cellPress", dgListener);
```

DataGrid.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se ha seleccionado un elemento.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando un componente DataGrid distribuye un evento `change`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.change` tiene una propiedad adicional, `type`, cuyo valor es `"change"`. Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `dgListener` y se pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `change` del parámetro `evt_obj`. Cuando se difunde el evento `change`, se envía una sentencia `trace` al panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Configurar datos de ejemplo.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Crear un objeto detector.
var dgListener:Object = new Object();
dgListener.change = function(evt_obj:Object) {
 trace("The selection has changed to " + evt_obj.target.selectedIndex);
};

// Añadir detector.
my_dg.addEventListener("change", dgListener);
```

DataGrid.columnCount

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.columnCount

Descripción

Propiedad (sólo lectura); número de columnas que aparecen en pantalla.

Ejemplo

El siguiente ejemplo muestra el número total de columnas en el panel Salida. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Añadir columnas a la cuadrícula y añadir datos.
my_dg.addColumn("a");
my_dg.addColumn("b");

my_dg.addItem({a:"one", b:"two"});

// Obtener número de columnas de la cuadrícula.
var colCount_num:Number = my_dg.columnCount;
trace("Number of columns: "+colCount_num);
```

DataGrid.columnNames

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.columnNames

Descripción

Propiedad; matriz de los nombres de campo de cada elemento que aparecen en forma de columnas.

Ejemplo

El siguiente ejemplo muestra el nombre de la columna del panel Salida cuando se hace clic en el título. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(200, 100);
my_dg.columnNames = ["Name", "Description", "Price"];

var dgListener:Object = new Object();
dgListener.headerRelease = function (evt_obj:Object) {
 trace("You clicked on the \"" + my_dg.columnNames[evt_obj.columnIndex] +
  "\" column.");
}
my_dg.addEventListener("headerRelease", dgListener);
```

DataGrid.columnStretch

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.columnStretch = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("columnStretch", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario cambia el tamaño de una columna horizontalmente.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando un componente DataGrid distribuye un evento `columnStretch`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.columnStretch` tiene dos propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino. La primera posición es 0.

`type` Cadena `"columnStretch"`.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

El siguiente ejemplo muestra el número de índice de la columna del panel Salida cuando se cambia el tamaño del título. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(240, 100);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({id:0, name:"Clark", score:3135});
myDP_array.push({id:1, name:"Bruce", score:403});
myDP_array.push({id:2, name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

// Crear un objeto detector.
var dgListener:Object = new Object();
dgListener.columnStretch = function(evt_obj:Object) {
    trace("column " + evt_obj.columnIndex + " was resized");
};

// Añadir detector.
my_dg.addEventListener("columnStretch", dgListener);
```

DataGrid.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.dataProvider

Descripción

Propiedad; modelo de datos de los elementos que se ven en el componente DataGrid.

La cuadrícula de datos añade métodos al prototipo de la clase Array de modo que cada objeto Array se adapta a la API DataProvider (véase el archivo DataProvider.as en la carpeta Classes/mx/controls/listclasses). Una matriz que se encuentre en el mismo fotograma o pantalla en forma de cuadrícula de datos tendrá todos los métodos (`addItem()`, `getItemAt()`, etc.) necesarios para que dicha cuadrícula sea el modelo de datos de una cuadrícula de datos. Asimismo, puede utilizarse para difundir cambios del modelo de datos en varios componentes.

En el componente DataGrid debe especificar los campos que aparecerán en la propiedad `DataGrid.columnNames`. Si no define la columna que se ha establecido (estableciendo la propiedad `DataGrid.columnNames` o llamando a `DataGrid.addColumn()`) para la cuadrícula de datos antes de que se haya establecido la propiedad `DataGrid.dataProvider`, dicha cuadrícula generará columnas para cada campo en el primer elemento del proveedor de datos, una vez que haya llegado dicho elemento.

Cualquier objeto que implemente la API DataProvider puede utilizarse como un proveedor de datos para una cuadrícula de datos (incluidos las matrices, los juegos de datos y los juegos de registros de Flash Remoting). Por ejemplo, véase "DataSet.dataProvider" en la página 370.

Use un proveedor de datos de la cuadrícula para la comunicación con los datos de la cuadrícula; el proveedor de datos funcionará de forma coherente, a pesar de la posición de desplazamiento.

Ejemplo

En el ejemplo siguiente se crea una matriz que se utilizará como proveedor de datos y se le asigna directamente la propiedad `dataProvider`:

```
my_dg.dataProvider = [{name:"Chris", price:"Priceless"}, {name:"Nigel",
  price:"cheap"}];
```

En el ejemplo siguiente se crea un nuevo objeto Array con la API DataProvider. Utiliza un bucle `for` para añadir 20 elementos a la cuadrícula:

```
var myDP:Array = new Array();
for (var i=0; i<20; i++)
  myDP.addItem({id:i, name:"Dave", price:"Priceless"});
my_dg.dataProvider = myDP
```

DataGrid.editable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.editable

Descripción

Propiedad; determina si el usuario puede editar la cuadrícula de datos (`true`) o no (`false`). Esta propiedad debe ser `true` para que puedan editarse columnas individuales y puedan seleccionarse celdas. El valor predeterminado es `false`.

Si desea no permitir la edición de columnas individuales, use la propiedad `DataGridColumn.editable`.

ATENCIÓN	El componente DataGrid no se podrá editar ni ordenar si se vincula directamente a un componente WebServiceConnector o XMLConnector. Debe vincular el componente DataGrid con el componente DataSet y vincular el componente DataSet con el componente WebServiceConnector o XMLConnector si desea que la cuadrícula se pueda editar u ordenar. Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

El ejemplo siguiente permite a los usuarios editar todas las columnas de la cuadrícula excepto la primera. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);

// Añadir columnas a la cuadrícula y añadir datos.
my_dg.addColumn("a");
my_dg.addColumn("b");
my_dg.addItem({a:"one", b:1});
my_dg.addItem({a:"two", b:2});

// Hacer que DataGrid sea editable.
my_dg.editable = true;
// Hacer que la primera columna sea de sólo lectura.
my_dg.getColumnAt(0).editable = false;
```

Véase también

`DataGridColumn.editable`

DataGrid.editField()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.editField(*index, colName, data*)

Parámetros

index Índice de la celda de destino. El valor está basado en cero.

colName Cadena que indica el nombre de la columna (campo) que contiene la celda de destino.

data Valor que debe almacenarse en la celda de destino. Este parámetro puede ser cualquier tipo de datos.

Valor devuelto

Los datos que estaban en la celda.

Descripción

Método; reemplaza los datos de celda en la ubicación especificada y actualiza la cuadrícula de datos con el nuevo valor. En cualquier celda presente para ese valor se activará el método `setValue()`.

Ejemplo

El siguiente ejemplo coloca un valor en la cuadrícula de la primera fila de la primera columna (valor de índice `0`) cuando se hace clic en el botón. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `my_btn` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Crear un objeto detector.
var btnListener:Object = new Object();
btnListener.click = function() {
   //Reemplazar el primer campo con valores nuevos.
   my_dg.editField(0, "name", "Arthur");
};

// Añadir un detector de botón.
my_btn.addEventListener("click", btnListener);
```

DataGrid.focusedCell

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.focusedCell

Descripción

Propiedad; únicamente en el modo editable, instancia de objeto que define la celda que está seleccionada. El objeto debe tener los campos `columnIndex` e `itemIndex`, los cuales son números enteros que indican el índice de la columna y el elemento de la celda. El origen es (0,0). El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se establece la celda seleccionada en la segunda columna, decimoprimera fila (con el número "10" porque la primera fila es la "0"). Como no es posible acceder a las celdas hasta que haya terminado de dibujarse el componente DataGrid, utilice `UIObject.doLater()` para provocar una demora mediante la propiedad `focusedCell`:

```
// Crear un proveedor de datos con 3 columnas y 50 filas.
var myDP:Array = new Array();
for (var i=0; i<50; i++)
  myDP.addItem({id:i, name:"Dave", price:"Priceless"});

// Asignar el proveedor de datos a la instancia de DataGrid y establecerlo
  como editable.
my_dg.dataProvider = myDP;
my_dg.editable = true;

// Utilizar UIObject.doLater() en la línea de tiempo actual para llamar a la
  función después que la cuadrícula de datos haya establecido todas sus
  propiedades.
my_dg.doLater(this, "select");

function select() {
 my_dg.focusedCell = {columnIndex:1, itemIndex:10};
}
```

DataGrid.getColumnAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*)

Parámetros

index Índice del objeto DataGridColumn que debe devolverse. El valor está basado en cero.

Valor devuelto

Un objeto DataGridColumn.

Descripción

Método; obtiene una referencia para el objeto DataGridColumn en el índice especificado.

Ejemplo

En el ejemplo siguiente se coloca el objeto DataGridColumn en el índice 0 y se cambia el texto. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `my_btn` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Crear un objeto detector.
var btnListener:Object = new Object();
btnListener.click = function() {
  // Colocar columna en ubicación 0.
  var a_dgc = my_dg.getColumnAt(0);
  // Cambiar texto de encabezado.
  a_dgc.headerText = "c";
};

// Añadir un detector de botón.
my_btn.addEventListener("click", btnListener);
```

DataGrid.getColumnIndex()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnIndex(*columnName*)

Parámetros

columnName Cadena de nombre de una columna.

Valor devuelto

Un número que especifica el índice de la columna.

Descripción

Método; devuelve el índice de la columna especificado por el parámetro *columnName*.

Ejemplo

El ejemplo siguiente muestra el número de índice de la columna "score". Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(150, 100);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

var column_num:Number = my_dg.getColumnIndex("score");
trace("Column that has name of 'score': " + column_num);
```

DataGrid.headerHeight

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.headerHeight

Descripción

Propiedad; altura de la barra de encabezado de la cuadrícula de datos, en píxeles. El valor predeterminado es 20.

Ejemplo

En el siguiente ejemplo se define la altura de la barra de encabezado en 40. Con una instancia del componente DataGrid en el escenario denominada `my_dg`, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
// Definir atributos de cuadrícula.
my_dg.setSize(240, 100);
my_dg.spaceColumnsEqually();

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

my_dg.headerHeight = 40;
```

DataGrid.headerRelease

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.headerRelease = function(eventObject){
  // Introducir aquí el código propio.
}
myDataGridInstance.addEventListener("headerRelease", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se libera un encabezado de columna. Puede utilizar este evento con la propiedad `DataGridColumn.sortOnHeaderRelease` para que no se ordene automáticamente, sino que se realice según se desee.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando el componente DataGrid distribuye un evento `headerRelease`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `DataGrid.headerRelease` tiene dos propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino.

`type` Cadena `"headerRelease"`.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `myListener` y se pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `headerRelease` en el parámetro *`eventObject`*. Cuando se difunde el evento `headerRelease`, se envía una sentencia `trace` al panel Salida.

```
var myListener = new Object();
myListener.headerRelease = function(event) {
    trace("column " + event.columnIndex + " header was pressed");
};
grid.addEventListener("headerRelease", myListener);
```

En el siguiente ejemplo se cambia el sentido de ordenación mediante una columna. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
var my_dg:mx.controls.DataGrid;
my_dg.setSize(150, 100);
my_dg.spaceColumnsEqually();
var myListener:Object = new Object();
myListener.headerRelease = function(evt:Object) {
  trace("column "+evt.columnIndex+" header was pressed");
  trace("\t current sort order is: "+evt.target.sortDirection);
  trace("");
};
my_dg.addEventListener("headerRelease", myListener);

my_dg.addColumn("a");
my_dg.addColumn("b");
my_dg.addItem({a:'one', b:1});
my_dg.addItem({a:'two', b:2});
```

Puede establecer si la ordenación es ascendente o descendente accediendo a la propiedad `sortDirection`. La propiedad `sortDirection` es una cadena que puede tener el valor `ASC` o `DESC`.

DataGrid.hScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.hScrollPolicy

Descripción

Propiedad; especifica si la cuadrícula de datos tiene una barra de desplazamiento horizontal. Esta propiedad puede tener el valor `"on"`, `"off"` o `"auto"`. El valor predeterminado es `"off"`.

Si establece `hScrollPolicy` en `"off"`, las columnas se escalarán de forma proporcional y se adaptarán a la anchura finita.

NOTA	Esto es distinto que en el componente List, en el que no se puede establecer `hScrollPolicy` en `"auto"`.

Ejemplo

En el ejemplo siguiente se establece la política de desplazamiento horizontal en automática, que significa que la barra de desplazamiento horizontal aparece si es necesario mostrar todo el contenido:

```
my_dg.setSize(150, 100);

// Añadir columnas a la cuadrícula y añadir datos.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

my_dg.hScrollPolicy = "on";
```

DataGrid.removeAllColumns()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.removeAllColumns()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los objetos DataGridColumn de la cuadrícula de datos. La llamada a este método no afecta al proveedor de datos.

Se debe llamar a este método si se está configurando un nuevo proveedor de datos con campos distintos de los del proveedor de datos anterior y se desea borrar los campos mostrados.

Ejemplo

En el siguiente ejemplo se eliminan todos los objetos DataGridColumn del componente DataGrid al hacer clic en el botón. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `clear_button` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

this.createClassObject(mx.controls.Button, "clear_button", 20,
  {label:"Clear"});
clear_button.move(10, 10);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

var buttonListener:Object = new Object();
buttonListener.click = function (evt_obj:Object) {
 my_dg.removeAllColumns();
}
clear_button.addEventListener("click", buttonListener);
```

DataGrid.removeColumnAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.removeColumnAt(*index*)

Parámetros

index Índice de la columna que se va a eliminar.

Valor devuelto

Una referencia al objeto DataGridColumn eliminado.

Descripción

Método; elimina el objeto DataGridColumn en el índice especificado.

Ejemplo

En el siguiente ejemplo se elimina el primer objeto DataGridColumn cuando se hace clic en el botón. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `name_button` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

name_button.setSize(140, name_button.height);
name_button.move(10, 10);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

// Crear un objeto detector.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 my_dg.removeColumnAt(my_dg.getColumnIndex("name"));
 evt_obj.target.enabled = false;
};
// Añadir un detector de botón.
name_button.addEventListener("click", buttonListener);
```

DataGrid.replaceItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.replaceItemAt(*index*, *item*)

Parámetros

index Índice del elemento que se va a reemplazar.

item Objeto que es el valor de elemento que se va a utilizar como sustitución.

Valor devuelto

El valor anterior.

Descripción

Método; reemplaza el elemento de un índice especificado y actualiza la presentación de la cuadrícula.

Ejemplo

En el siguiente ejemplo se reemplaza el elemento en el índice de fila 2 por entradas nuevas. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `replace_button` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

replace_button.move(10, 10);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

// Crear un objeto detector.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 //Reemplazar valor anterior
 var prevValue_obj:Object = my_dg.replaceItemAt(2, {name:"Frank",
  score:949});
 my_dg.selectedIndex = 2;
};
// Añadir un detector de botón.
replace_button.addEventListener("click", buttonListener);
```

DataGrid.resizableColumns

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.resizableColumns

Descripción

Propiedad; valor booleano que determina si el usuario puede (`true`) o no puede (`false`) expandir las columnas de la cuadrícula. Esta propiedad debe ser `true` para que el usuario pueda cambiar el tamaño de columnas individuales. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se impide a los usuarios la posibilidad de cambiar el tamaño de las columnas. Con una instancia del componente DataGrid denominada `my_dg` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

// No permitir que se cambie el tamaño de las columnas.
my_dg.resizableColumns = false;
```

DataGrid.selectable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.selectable

Descripción

Propiedad; valor booleano que determina si el usuario puede seleccionar una cuadrícula de datos (true) o no (false). El valor predeterminado es true. Si se devuelve el valor false, un elemento de la cuadrícula no permanece seleccionado cuando el usuario hace clic en el elemento y mueve el puntero.

Ejemplo

En el ejemplo siguiente se impide seleccionar la cuadrícula. Con una instancia del componente DataGrid denominada my_dg en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

my_dg.selectable = false;
```

DataGrid.showHeaders

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.showHeaders

Descripción

Propiedad; valor booleano que determina si la cuadrícula de datos muestra los encabezados de columna (true) o no (false). Los encabezados de columna aparecen sombreados para distinguirlos de las demás filas de la cuadrícula. El usuario puede hacer clic en un encabezado de columna para ordenar el contenido de dicha columna si el valor de DataGrid.sortableColumns se establece en true. El valor predeterminado de showHeaders es true.

Ejemplo

En el ejemplo siguiente se ocultan los encabezados de columna:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No mostrar encabezados.
my_dg.showHeaders = false;
```

Véase también

`DataGrid.sortableColumns`

DataGrid.sortableColumns

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid`.sortableColumns`

Descripción

Propiedad; valor booleano que determina si las columnas de una cuadrícula de datos pueden ordenarse (`true`) o no (`false`) cuando el usuario hace clic en un encabezado de columna. Esta propiedad debe ser `true` para que se pueda ordenar columnas individuales y para que se difunda el evento `headerRelease`. El valor predeterminado es `true`.

ATENCIÓN	El componente DataGrid no se podrá editar ni ordenar si se vincula directamente a un componente WebServiceConnector o XMLConnector. Debe vincular el componente DataGrid con el componente DataSet y vincular el componente DataSet con el componente WebServiceConnector o XMLConnector si desea que la cuadrícula se pueda editar u ordenar. Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

En el ejemplo siguiente se desactiva la función de ordenación:

```
my_dg.setSize(140, 100);

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No permitir que se ordenen las columnas.
my_dg.sortableColumns = false;
```

Véase también

`DataGrid.headerRelease`

DataGrid.spaceColumnsEqually()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.spaceColumnsEqually()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; vuelve a espaciar las columnas de manera uniforme.

Ejemplo

En el siguiente ejemplo se vuelven a espaciar las columnas de `my_dg` al hacer clic en el botón. Con una instancia del componente DataGrid denominada `my_dg` y una instancia del componente Button denominada `resize_button` en el escenario, pegue el siguiente código en el primer fotograma de la línea de tiempo principal:

```
my_dg.move(10, 40);
my_dg.setSize(200, 100);

resize_button.move(10, 10);
resize_button.setSize(200, resize_button.height);

my_dg.addColumn("guitar");
my_dg.addColumn("name");

// Configurar datos de ejemplo.
my_dg.addItem({guitar:"Flying V", name:"maggot"});
my_dg.addItem({guitar:"SG", name:"dreschie"});
my_dg.addItem({guitar:"jagstang", name:"vitapup"});

// Crear un objeto detector.
var buttonListener:Object = new Object();
buttonListener.click = function() {
 my_dg.spaceColumnsEqually();
};
// Añadir un detector de botón.
resize_button.addEventListener("click", buttonListener);
```

Clase DataGridColumn (sólo en Flash Professional)

Nombre de clase de ActionScript mx.controls.gridclasses.DataGridColumn

Se pueden crear y configurar objetos DataGridColumn para utilizarlos como columnas de una cuadrícula de datos. Muchos de los métodos de la clase DataGrid tienen la función de administrar objetos DataGridColumn. Los objetos DataGridColumn se almacenan en una matriz basada en cero en la cuadrícula de datos, donde 0 indica la columna situada en el extremo izquierdo. Después de añadir o crear columnas, puede acceder a dichas columnas llamando a `DataGrid.getColumnAt(`*`index`*`)`.

Existen tres formas de añadir o crear columnas en una cuadrícula. Si desea configurar las columnas, es mejor utilizar el segundo o el tercer método antes de añadir datos a la cuadrícula, de manera que no haya que crear las columnas dos veces.

- Añada un proveedor de datos o un elemento con varios campos a una cuadrícula en la que no se haya configurado ningún objeto DataGridColumn. De este modo, se generarán automáticamente columnas en cada campo en orden inverso al del bucle `for..in`. Por ejemplo, para una instancia de DataGrid denominada `my_dg`:

```
my_dg.dataProvider = [{guitar:"Flying V", name:"maggot"}, {guitar:"SG",
  name:"dreschie"}, {guitar:"jagstang", name:"vitapup"}];
```

- Utilice `DataGrid.columnNames` para crear los nombres de campo de los campos de elemento deseados y generar en orden objetos DataGridColumn para cada uno de los campos enumerados. Este sistema permite seleccionar y ordenar columnas rápidamente, lo cual reduce al mínimo las necesidades de configuración. Además, se elimina la información anterior de las columnas. Por ejemplo, para una instancia de DataGrid denominada `my_dg`:

```
my_dg.columnNames = ["guitar","name"];
```

- Genere previamente los objetos DataGridColumn y añádalos a la cuadrícula de datos mediante `DataGrid.addColumn()`. Este método es útil, y el más flexible, porque permite añadir columnas con el tamaño y formato adecuados antes de que las columnas se incluyan en la cuadrícula (con lo que se reduce el consumo de recursos del procesador). Para más información, consulte "Constructor de la clase DataGridColumn" en la página 317. Por ejemplo, para una instancia de DataGrid denominada `my_dg`:

```
// Crear un objeto de columna.
var location_dgc:DataGridColumn = new DataGridColumn("Location");
location_dgc.width = 100;
// Añadir una columna a DataGrid.
my_dg.addColumn(location_dgc);
```

Resumen de propiedades de la clase DataGridColumn

En la tabla siguiente se enumeran las propiedades de la clase DataGridColumn.

Propiedad	Descripción
`DataGridColumn.cellRenderer`	Identificador de la vinculación de un símbolo que se va a utilizar para mostrar las celdas de la columna.
`DataGridColumn.columnName`	Sólo lectura; el nombre del campo asociado a la columna.

Propiedad	Descripción
DataGridColumn.editable	Valor booleano que indica si la columna es editable (true) o no (false).
DataGridColumn.headerRenderer	Nombre de una clase que se utilizará para mostrar el encabezado de la columna.
DataGridColumn.headerText	Texto del encabezado de la columna.
DataGridColumn.labelFunction	Función que determina qué campo de un elemento va a mostrarse.
DataGridColumn.resizable	Valor booleano que indica si puede cambiarse el tamaño de la columna (true) o no (false).
DataGridColumn.sortable	Valor booleano que indica si la columna puede ordenarse (true) o no (false).
DataGridColumn.sortOnHeaderRelease	Valor booleano que indica si la columna está ordenada (true) o no (false) cuando un usuario hace clic en un encabezado de columna.
DataGridColumn.width	Anchura de la columna, expresada en píxeles.

Constructor de la clase DataGridColumn

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
new DataGridColumn(name)
```

Parámetros

name Cadena que indica el nombre del objeto DataGridColumn. Este parámetro corresponde al campo que se mostrará de cada elemento.

Valor devuelto

Ninguno.

Descripción

Constructor; crea un objeto DataGridColumn. Utilice este constructor para crear columnas y añadirlas al componente DataGrid. Después de crear objetos DataGridColumn podrá añadirlos a la cuadrícula de datos mediante una llamada a `DataGrid.addColumn()`.

Ejemplo

En el ejemplo siguiente se crea un objeto DataGridColumn denominado `Location`:

```
import mx.controls.gridclasses.DataGridColumn;
var column = new DataGridColumn("Location");
```

DataGridColumn.cellRenderer

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid`.getColumnAt(`*index*`).cellRenderer`

Descripción

Propiedad; identificador de la vinculación de un símbolo que se va a utilizar para mostrar las celdas de la columna. Todas las clases que se utilicen con esta propiedad deben implementar la API CellRenderer (véase "Interfaz API CellRenderer" en la página 113). El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de celdas:

```
myGrid.getColumnAt(3).cellRenderer = "MyCellRenderer";
```

DataGridColumn.columnName

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).columnName

Descripción

Propiedad (sólo lectura); nombre del campo asociado con la columna. El valor predeterminado es el nombre al que se llama en el constructor DataGridColumn.

Ejemplo

En el ejemplo siguiente se muestra el nombre de la columna como posición de índice 1:

```
import mx.controls.gridclasses.DataGridColumn;
// Definir atributos de cuadrícula.
my_dg.setSize(150, 100);

// Añadir columnas a la cuadrícula.
var name_dgc:DataGridColumn = my_dg.addColumn(new DataGridColumn("name"));
name_dgc.headerText = "Name:";
var score_dgc:DataGridColumn = my_dg.addColumn(new
  DataGridColumn("score"));
score_dgc.headerText = "Score:";

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Obtener nombre de columna.
var name_str:String = my_dg.getColumnAt(1).columnName;
trace(name_str);
```

Véase también

Constructor de la clase DataGridColumn

DataGridColumn.editable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).editable

Descripción

Propiedad; determina si el usuario puede editar la columna (`true`) o no (`false`). La propiedad `DataGrid.editable` debe tener un valor de `true` para que puedan editarse las distintas columnas, incluso cuando el valor de `DataGridColumn.editable` sea `true`. El valor predeterminado es `true`.

ATENCIÓN

El componente DataGrid no se podrá editar ni ordenar si se vincula directamente a un componente WebServiceConnector o XMLConnector. Debe vincular el componente DataGrid con el componente DataSet y vincular el componente DataSet con el componente WebServiceConnector o XMLConnector si desea que la cuadrícula se pueda editar u ordenar. Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

En el ejemplo siguiente se impide que se puedan editar los elementos de la primera columna de una cuadrícula:

```
// Definir atributos de cuadrícula.
my_dg.setSize(150, 100);
my_dg.editable = true;

// Añadir columnas a la cuadrícula.
my_dg.addColumn("name");
my_dg.addColumn("score");

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No permitir que la primera columna sea editable.
my_dg.getColumnAt(0).editable = false;
```

Véase también

`DataGrid.editable`

DataGridColumn.headerRenderer

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).headerRenderer

Descripción

Propiedad; cadena que indica el nombre de clase que se utilizará para mostrar el encabezado de la columna. Todas las clases que se utilicen con esta propiedad deben implementar la API CellRenderer (véase "Interfaz API CellRenderer" en la página 113). El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de encabezados:

```
myGrid.getColumnAt(3).headerRenderer = "MyHeaderRenderer";
```

DataGridColumn.headerText

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).headerText

Descripción

Propiedad; texto del encabezado de la columna. El valor predeterminado es el nombre de la columna.

Esta propiedad permite mostrar algo más que el nombre del campo como encabezado.

Ejemplo

En el ejemplo siguiente se define "Price (USD)" como texto de encabezado de la columna:

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;

var price_dgc:DataGridColumn = new DataGridColumn("price");
price_dgc.headerText = "Price (USD)";
price_dgc.width = 80;
my_dg.addColumn(price_dgc);

my_dg.addItem({price:"$14.99"});
```

DataGridColumn.labelFunction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).labelFunction

Descripción

Propiedad; especifica una función para determinar qué campo (o combinación de campos) de cada elemento va a mostrarse. Esta función recibe un parámetro *item*, que es el elemento que se va a representar, y debe devolver una cadena que represente el texto que va a mostrarse. Esta propiedad puede utilizarse para crear columnas virtuales que no tengan campos equivalentes en el elemento.

NOTA	La función especificada opera en un ámbito no definido.

Ejemplo

En el ejemplo siguiente se calcula un valor para la columna "Subtotal":

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(300, 200);

// Configurar columnas.
var guitar_dgc:DataGridColumn = new DataGridColumn("guitar");
var value_dgc:DataGridColumn = new DataGridColumn("value");
var tax_dgc:DataGridColumn = new DataGridColumn("tax");
var st_dgc:DataGridColumn = new DataGridColumn("Subtotal");
// Definir labelFunction para columna Subtotal.
st_dgc.labelFunction = function(item:Object):String {
 if ((item.value != undefined) && (item.tax != undefined)) {
  return "$"+(item.value+item.tax);
 }
};

// Añadir columnas a la cuadrícula.
my_dg.addColumn(guitar_dgc);
my_dg.addColumn(value_dgc);
my_dg.addColumn(tax_dgc);
my_dg.addColumn(st_dgc);
```

```
// Definir modelo de datos.
my_dg.addItem({guitar:"Flying V", value:10, tax:1});
my_dg.addItem({guitar:"SG", value:20, tax:2});
my_dg.addItem({guitar:"jagstang", value:30, tax:3});
```

DataGridColumn.resizable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).resizable

Descripción

Propiedad; valor booleano que determina si el usuario puede cambiar el tamaño de la columna (`true`) o no (`false`). Para que esta propiedad pueda aplicarse, el valor de la propiedad `DataGrid.resizableColumns` debe establecerse en `true`. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se impide la posibilidad de cambiar el tamaño de la columna del índice 0:

```
// Definir atributos de cuadrícula.
my_dg.setSize(150, 100);
my_dg.addColumn("name");
my_dg.addColumn("score");

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No permitir que se cambie el tamaño de la primera columna.
my_dg.getColumnAt(0).resizable = false;
```

DataGridColumn.sortable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDataGrid.getColumnAt(index).sortable
```

Descripción

Propiedad; valor booleano que indica si el usuario puede ordenar una columna (`true`) o no (`false`). Para que esta propiedad pueda aplicarse, el valor de la propiedad `DataGrid.sortableColumns` debe establecerse en `true`. El valor predeterminado es `true`.

ATENCIÓN	El componente DataGrid no se podrá editar ni ordenar si se vincula directamente a un componente WebServiceConnector o XMLConnector. Debe vincular el componente DataGrid con el componente DataSet y vincular el componente DataSet con el componente WebServiceConnector o XMLConnector si desea que la cuadrícula se pueda editar u ordenar. Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

En el ejemplo siguiente se impide la posibilidad de ordenar la columna del índice 1:

```
// Definir atributos de cuadrícula.
my_dg.setSize(150, 100);

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No permitir que se ordene la segunda columna.
my_dg.getColumnAt(1).sortable = false;
```

DataGridColumn.sortOnHeaderRelease

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDataGrid.getColumnAt(*index*).sortOnHeaderRelease

Descripción

Propiedad; valor booleano que indica si la columna se ordenará automáticamente cuando el usuario haga clic en el encabezado (`true`) o no (`false`). El valor de esta propiedad sólo puede ser `true` si el de `DataGridColumn.sortable` es `true`. Si el valor de `DataGridColumn.sortOnHeaderRelease` está establecido en `false`, el usuario puede capturar el evento `headerRelease` para realizar su propia ordenación.

El valor predeterminado es `true`.

ATENCIÓN	El componente DataGrid no se podrá editar ni ordenar si se vincula directamente a un componente WebServiceConnector o XMLConnector. Debe vincular el componente DataGrid con el componente DataSet y vincular el componente DataSet con el componente WebServiceConnector o XMLConnector si desea que la cuadrícula se pueda editar u ordenar. Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

En el siguiente ejemplo se desactiva la ordenación de la segunda columna:

```
// Definir atributos de cuadrícula.
my_dg.setSize(150, 100);

// Configurar datos de ejemplo.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// No ordenar la segunda columna al hacer clic en el encabezado.
my_dg.getColumnAt(1).sortOnHeaderRelease = false;
```

DataGridColumn.width

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDataGrid.getColumnAt(index).width
```

Descripción

Propiedad; número que indica la anchura de la columna, expresada en píxeles. El valor predeterminado es 50.

Ejemplo

En el siguiente ejemplo se aplica a la primera columna una anchura de 50 píxeles:

```
// Crear nuevo componente DataProvider.
var myDP_array:Array = new Array({name:"Chris", price:"Priceless"},
  {name:"Nigel", price:"Cheap"});
// Asignar DataProvider a DataGrid.
my_dg.dataProvider = myDP_array;

// Modificar dimensiones de DataGrid.
my_dg.setSize(140, 100);
my_dg.rowHeight = 30;
my_dg.getColumnAt(0).width = 50;
```

CAPÍTULO 11

Componente DataHolder (sólo en Flash Professional)

11

El componente DataHolder es un depósito para datos y un medio de generar eventos cuando se modifican dichos datos. Su función principal consiste en almacenar datos y actuar como conector entre otros componentes que utilizan vinculación de datos.

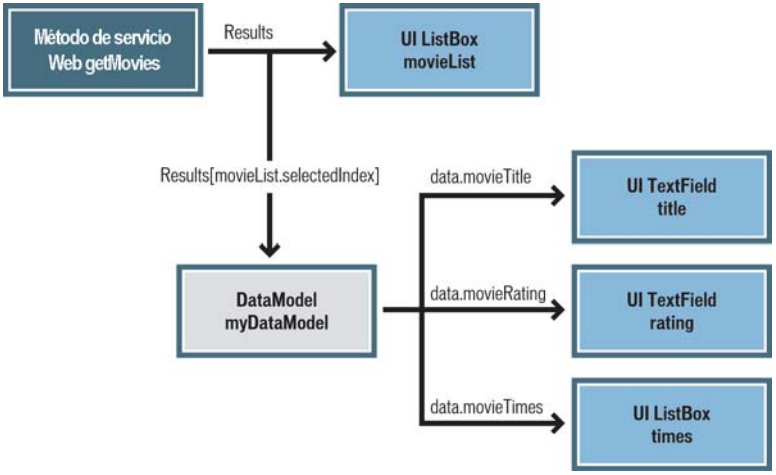
Inicialmente, el componente DataHolder tiene una única propiedad que puede vincularse denominada `data`. Es posible añadir más propiedades con la ficha Esquema del inspector de componentes. Para más información sobre el uso de la ficha Esquema, consulte "Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)" en *Utilización de Flash*.

El usuario puede asignar cualquier tipo de datos a una propiedad DataHolder, ya sea creando una vinculación entre los datos y otra propiedad o utilizando su propio código ActionScript. Cuando el valor de los datos cambia, el componente DataHolder emite un evento cuyo nombre coincide con el de la propiedad y se ejecutan las vinculaciones asociadas con dicha propiedad.

En la mayoría de los casos, no utilizará este componente para crear una aplicación. Sólo es necesario cuando no pueda vincular datos externos directamente a otro componente y no desee utilizar un componente DataSet. El componente DataHolder resulta de utilidad cuando no es posible vincular componentes directamente entre sí (por ejemplo conectores, componentes de interfaz de usuario o componentes DataSet). A continuación se muestran algunos supuestos en los que se utiliza un componente DataHolder:

- Si un valor de datos se genera mediante ActionScript, es posible que haya que vincularlo a otros componentes. En este caso, puede obtenerse un componente DataHolder que contenga propiedades vinculadas de la forma deseada. Cuando se asignen nuevos valores a esas propiedades (mediante ActionScript, por ejemplo), dichos valores se distribuirán al objeto que esté vinculado a los datos.

- Es posible que exista un valor de datos resultante de una vinculación de datos indexada compleja, como se muestra en el diagrama siguiente.



En este caso, es conveniente vincular el valor de datos a un componente DataHolder (denominado *DataModel* en esta ilustración) y, a continuación, utilizarlo para realizar vinculaciones a la interfaz de usuario.

NOTA	El componente DataHolder no está pensado para tener el mismo tipo de control sobre los datos que el componente DataSet. No administra ni rastrea datos y tampoco puede actualizarlos. Es un repositorio para almacenar datos y generar eventos cuando se modifican estos datos.

Creación de una aplicación con el componente DataHolder (sólo en Flash Professional)

En este ejemplo se añade una propiedad de matriz al esquema del componente DataHolder (una matriz) cuyo valor lo determina el código ActionScript escrito por el usuario. Después se podrá vincular esa propiedad de matriz a la propiedad `dataProvider` de un componente DataGrid utilizando la ficha Vinculaciones del inspector de componentes.

Para utilizar el componente DataHolder en una aplicación sencilla:

1. En Flash, cree un nuevo archivo.
2. Abra el panel Componentes, arrastre un componente DataHolder al escenario y asígnele el nombre **dataHolder**.
3. Arrastre un componente DataGrid al escenario y asígnele el nombre **namesGrid**.
4. Seleccione el componente DataHolder y abra el inspector de componentes.
5. Haga clic en la ficha Esquema del inspector de componentes.
6. Haga clic en el botón Añadir una propiedad de componente (+) situado en el panel superior de la ficha Esquema.
7. En el panel inferior de la ficha Esquema, escriba **namesArray** en el campo de nombre y seleccione Array en el menú emergente de tipo de datos.
8. Haga clic en la ficha Vinculaciones del inspector de componentes y añada una vinculación entre la propiedad `namesArray` del componente DataHolder y la propiedad `dataProvider` del componente DataGrid.

   Para más información sobre la creación de vinculaciones con la ficha Vinculaciones, consulte “Trabajo con vinculaciones en la ficha Vinculaciones (sólo Flash Professional)” en *Utilización de Flash*.
9. En la línea de tiempo, seleccione el primer fotograma de la capa 1 y abra el panel Acciones.
10. Introduzca el código siguiente en el panel Acciones:

    ```
    dataHolder.namesArray = [{name:"Tim"},{name:"Paul"},{name:"Jason"}];
    ```

    Con este código se llena la matriz `namesArray` con varios objetos. Cuando se ejecuta esta asignación de variables, se ejecuta también la vinculación que se estableció con anterioridad entre el componente DataHolder y el componente DataGrid.
11. Para probar el archivo, seleccione Control > Probar película.

Clase DataHolder

Herencia MovieClip > DataHolder

Nombre de clase de ActionScript mx.data.components.DataHolder

El componente DataHolder es un repositorio para datos y un medio de generar eventos cuando se modifican dichos datos. Su función principal consiste en almacenar datos y actuar como conector entre otros componentes que utilizan vinculación de datos.

Inicialmente, el componente DataHolder tiene una única propiedad que puede vincularse denominada `data`. Es posible añadir más propiedades con la ficha Esquema del inspector de componentes.

Resumen de propiedades de la clase DataHolder

En la siguiente tabla se enumeran las propiedades de la clase DataHolder.

Propiedad	Descripción
`DataHolder.data`	Propiedad vinculable predeterminada del componente DataHolder.

DataHolder.data

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`dataHolder``.data`

Descripción

Propiedad; elemento predeterminado del esquema de un objeto DataHolder. Esta propiedad no es una parte "permanente" del componente DataHolder. Se trata, por el contrario, de una propiedad vinculable predeterminada para cada instancia del componente. El usuario puede añadir propiedades vinculables propias o eliminar la propiedad `data` predeterminada utilizando la ficha Esquema del inspector de componentes.

Para más información sobre el uso de la ficha Esquema, consulte "Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)" en *Utilización de Flash*.

Ejemplo

Para ver un ejemplo paso a paso de cómo utilizar este componente, consulte "Creación de una aplicación con el componente DataHolder (sólo en Flash Professional)" en la página 329.

En el código siguiente se muestra un ejemplo simple de cómo rellenar el componente DataHolder con datos que son variables. Para probar la aplicación, es necesario introducir un valor en el campo de entrada de texto y hacer clic en la instancia `addDate_btn`, que añade el valor al componente DataHolder. Haga clic en la instancia `dumpDataHolder_btn` para rastrear el contenido del componente DataHolder.

```
// Arrastrar dos componentes Button al escenario (addDate_btn y
  dumpDataHolder_btn), un TextInput (myDate_txt) y un DataHolder
  (myDataHolder). Añadir el siguiente código ActionScript al fotograma 1:

var dhListener:Object = new Object();
dhListener.click = function() {
  trace("dumping DataHolder");
  trace("  " + myDataHolder.myDate);
  trace("");
};
var dateListener:Object = new Object();
dateListener.click = function() {
  myDataHolder.myDate = myDate_txt.text;
  trace("added value");
};
this.dumpDataHolder_btn.addEventListener("click", dhListener);
this.addDate_btn.addEventListener("click", dateListener);
```

CAPÍTULO 12

Interfaz API de DataProvider 12

La interfaz API de DataProvider es un conjunto de métodos y propiedades que un origen de datos debe tener para que una clase basada en listas pueda comunicarse con dicho origen. Las matrices, juegos de registros y juegos de datos implementan esta interfaz API. Se puede crear una clase compatible con DataProvider si se implementan todos los métodos y propiedades descritos en esta sección. Un componente basado en listas podría entonces utilizar esa clase como proveedor de datos.

Clase DataProvider

Nombre de clase de ActionScript mx.controls.listclasses.DataProvider

Los métodos de la clase DataProvider permiten consultar y modificar datos de los componentes que muestren datos (también denominados *vistas*). La interfaz API de DataProvider también difunde eventos `change` si cambian los datos. Distintas vistas pueden utilizar el mismo proveedor de datos y recibir eventos `change`.

Un proveedor de datos es una colección lineal (como una matriz) de elementos. Cada elemento es un objeto compuesto de muchos campos de datos. Se puede acceder a cada elemento a través del índice (tal como se haría en una matriz) mediante `DataProvider.getItemAt()`.

Los proveedores de datos suelen utilizarse con matrices. Los componentes para datos aplican todos los métodos de la interfaz API de DataProvider a `Array.prototype` cuando hay un objeto Array en el mismo fotograma o en la misma pantalla donde está el componente para datos. De esta forma se puede utilizar una matriz existente como datos para vistas que tengan una propiedad `dataProvider`.

Gracias a la interfaz API de DataProvider, los componentes de la versión 2 de la arquitectura de componentes de Macromedia que permiten obtener vistas de datos (DataGrid, List, Tree, etc.) también pueden mostrar objetos RecordSet de Flash Remoting y datos del componente DataSet. La interfaz API de DataProvider es el lenguaje con el que los componentes para datos se comunican con sus proveedores de datos.

En la documentación de Macromedia Flash, “DataProvider” es el nombre de la clase, `dataProvider` es una propiedad de cada componente que permite obtener una vista de los datos y “proveedor de datos” es la expresión genérica que se emplea para indicar el origen de datos.

Resumen de métodos de la interfaz API de DataProvider

En la siguiente tabla se enumeran los métodos de la interfaz API de DataProvider.

Método	Descripción
`DataProvider.addItem()`	Añade un elemento al final del proveedor de datos.
`DataProvider.addItemAt()`	Añade un elemento al proveedor de datos en la posición especificada.
`DataProvider.editField()`	Cambia un campo del proveedor de datos.
`DataProvider.getEditingData()`	Obtiene los datos para su edición desde un proveedor de datos.
`DataProvider.getItemAt()`	Obtiene una referencia para el elemento en una posición especificada.
`DataProvider.getItemID()`	Devuelve el ID exclusivo del elemento.
`DataProvider.removeAll()`	Elimina todos los elementos de un proveedor de datos.
`DataProvider.removeItemAt()`	Elimina un elemento del proveedor de datos en una posición especificada.
`DataProvider.replaceItemAt()`	Sustituye por otro el elemento de una posición especificada.
`DataProvider.sortItems()`	Ordena los elementos en el proveedor de datos según una función de comparación u opciones de clasificación.
`DataProvider.sortItemsBy()`	Ordena los elementos del proveedor de datos alfabética o numéricamente, en el orden especificado, mediante el nombre de campo especificado.

Resumen de propiedades de la interfaz API de DataProvider

En la siguiente tabla se enumeran las propiedades de la interfaz API de DataProvider.

Propiedad	Descripción
`DataProvider.length`	Número de elementos de un proveedor de datos.

Resumen de eventos de la interfaz API de DataProvider

En la siguiente tabla se enumeran los eventos de la interfaz API de DataProvider.

Evento	Descripción
`DataProvider.modelChanged`	Se difunde cuando cambia el proveedor de datos.

DataProvider.addItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP.addItem(*item*)

Parámetros

item Objeto que contiene datos. Constituye un elemento de un proveedor de datos.

Valor devuelto

Ninguno.

Descripción

Método; añade un elemento nuevo al final del proveedor de datos. Este método activa el evento `modelChanged` con el nombre de evento `addItems`.

Ejemplo

En el ejemplo siguiente se añade un elemento al final del proveedor de datos `myDP`:

```
myDP.addItem({label : "this is an Item"});
```

DataProvider.addItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.addItemAt(index, item)
```

Parámetros

index Número igual o mayor que 0. Este número indica la posición en la que se debe insertar el elemento; es el índice del nuevo elemento.

item Objeto que contiene los datos del elemento.

Valor devuelto

Ninguno.

Descripción

Método; añade un nuevo elemento al proveedor de datos en el índice especificado. Se omiten los índices cuya longitud sea mayor que la del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `addItems`.

Ejemplo

En el ejemplo siguiente se añade un elemento al proveedor de datos `myDP` en la cuarta posición:

```
myDP.addItemAt(3, {label : "this is the fourth Item"});
```

DataProvider.editField()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.editField(index, fieldName, newData)
```

Parámetros

index Número mayor o igual que 0; el índice del elemento.

fieldName Cadena que indica el nombre de campo del elemento que se modificará.

newData Nuevos datos que se colocarán en el proveedor de datos.

Valor devuelto

Ninguno.

Descripción

Método; cambia un campo del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `updateField`.

Ejemplo

El código siguiente modifica el campo `label` del tercer elemento:

```
myDP.editField(2, "label", "mynewData");
```

DataProvider.getEditingData()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.getEditingData(index, fieldName)
```

Parámetros

index Número mayor o igual que 0 y menor que `DataProvider.length`. Este número es el índice del elemento que debe recuperarse.

fieldName Cadena que indica el nombre del campo que se va a editar.

Valor devuelto

Los datos con formato editables que se utilizarán.

Descripción

Método; recupera los datos para su edición desde un proveedor de datos. De esta forma el modelo de datos puede proporcionar distintos formatos de datos para su edición y visualización.

Ejemplo

Con el código siguiente se obtiene una cadena editable para el campo de precio:

```
trace(myDP.getEditingData(4, "price");
```

DataProvider.getItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP.getItemAt(*index*)

Parámetros

index Número mayor o igual que 0 y menor que `DataProvider.length`. Este número es el índice del elemento que debe recuperarse.

Valor devuelto

Una referencia al elemento recuperado; no definido si el índice está fuera de rango.

Descripción

Método; recupera una referencia para el elemento en una posición especificada.

Ejemplo

En el código siguiente se muestra la etiqueta del quinto elemento:

```
trace(myDP.getItemAt(4).label);
```

DataProvider.getItemID()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP.getItemID(*index*)

Parámetros

index Número mayor o igual que 0.

Valor devuelto

Un número que es el ID exclusivo del elemento.

Descripción

Método; devuelve un ID exclusivo del elemento. Este método se emplea principalmente en el seguimiento de una selección. Este ID se utiliza en componentes para datos con el fin de conservar listas de los elementos que están seleccionados.

Ejemplo

En este ejemplo se obtiene el ID del cuarto elemento:

```
var ID = myDP.getItemID(3);
```

DataProvider.length

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP.length

Descripción

Propiedad (sólo lectura); número de elementos del proveedor de datos.

Ejemplo

En este ejemplo se envía al panel Salida una cantidad de elementos equivalente al número especificado en el proveedor de datos `myArray`:

```
trace(myArray.length);
```

DataProvider.modelChanged

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.modelChanged = function(eventObject){
  // Introducir aquí el código propio.
}
myMenu.addEventListener("modelChanged", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores de vistas cuando se modifica el proveedor de datos. Normalmente, un detector se añade a un modelo mediante la asignación de su propiedad `dataProvider`.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. Cuando el proveedor de datos experimenta algún cambio, difunde un evento `modelChanged` y los componentes para datos lo capturan para actualizar sus vistas, de forma que queden reflejados los cambios que hayan sufrido los datos.

El objeto del evento `Menu.modelChanged` tiene cinco propiedades adicionales:

- `eventName` La propiedad `eventName` se utiliza para dividir los eventos `modelChanged` en subcategorías. Los componentes para datos utilizan esta información para evitar tener que actualizar completamente la instancia (vista) del componente que esté utilizando el proveedor de datos. La propiedad `eventName` admite los siguientes valores:
  - `updateAll` Es necesario actualizar toda la vista, excluida la posición de desplazamiento.
  - `addItems` Se ha añadido una serie de elementos.
  - `removeItems` Se ha eliminado una serie de elementos.
  - `updateItems` Es necesario actualizar una serie de elementos.
  - `sort` Se han ordenado los datos.
  - `updateField` Un campo de un elemento debe cambiarse y actualizarse.
  - `updateColumn` Es necesario actualizar la definición de la totalidad de un campo del proveedor de datos.

    - `filterModel` El modelo se ha filtrado y es necesario actualizar la vista (restablecer la posición de desplazamiento).
    - `schemaLoaded` Se ha declarado la definición del campo del proveedor de datos.
- `firstItem` Índice del primer elemento afectado.
- `lastItem` Índice del último elemento afectado. El valor es igual al de `firstItem` si se ve afectado un único elemento.
- `removedIDs` Matriz de los identificadores de elemento que se han eliminado.
- `fieldName` Cadena que indica el nombre del campo que se verá afectado.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el ejemplo siguiente, se define un controlador denominado `listener` y luego se pasa a `addEventListener()` como segundo parámetro. El controlador `modelChanged` captura el objeto de evento en el parámetro `evt`. Cuando se difunde el evento `modelChanged`, se envía una sentencia `trace` al panel Salida.

```
listener = new Object();
listener.modelChanged = function(evt){
    trace(evt.eventName);
}
myList.addEventListener("modelChanged", listener);
```

DataProvider.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP`.removeAll()`

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos del proveedor de datos. Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

Ejemplo

En este ejemplo se eliminan todos los elementos del proveedor de datos:

```
myDP.removeAll();
```

DataProvider.removeItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myDP.removeItemAt(*index*)

Parámetros

index Número mayor o igual que 0. Este número es el índice del elemento que se va a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina el elemento del índice especificado. Los índices que se encuentran después del índice eliminado se contraerán una posición.

Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

Ejemplo

En este ejemplo se elimina el elemento de la cuarta posición:

```
myDP.removeItemAt(3);
```

DataProvider.replaceItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.replaceItemAt(index, item)
```

Parámetros

index Número mayor o igual que 0. Este número es el índice del elemento que se va a cambiar.

item Objeto que es el nuevo elemento.

Valor devuelto

Ninguno.

Descripción

Método; sustituye el contenido del elemento en el índice especificado. Este método activa el evento `modelChanged` con el nombre de evento `updateItems`.

Ejemplo

En este ejemplo se sustituye el elemento del índice 3 por el elemento que tiene la etiqueta "new label":

```
myDP.replaceItemAt(3, {label : "new label"});
```

DataProvider.sortItems()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.sortItems([compareFunc], [optionsFlag])
```

Parámetros

compareFunc Referencia a una función que se utiliza para comparar dos elementos y determinar su orden de clasificación. Para más información, consulte %{sort (método Array.sort)}% en *Referencia del lenguaje ActionScript 2.0*. Este parámetro es opcional.

optionsFlag Permite realizar muchos tipos distintos de ordenaciones en una única matriz sin necesidad de duplicar toda la matriz ni de volver a ordenarla repetidas veces. Este parámetro es opcional.

Los valores posibles de *optionsFlag* son:

- `Array.DESCENDING`, que ordena de mayor a menor.
- `Array.CASEINSENSITIVE`, que ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`, que ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas, que se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado esa etiqueta.
- `Array.UNIQUESORT`, que devuelve un código de error (0) en lugar de una matriz ordenada si dos objetos de la matriz son idénticos o lo son sus campos de ordenación.
- `Array.RETURNINDEXEDARRAY`, que devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, la siguiente matriz devolvería la segunda línea de código y la matriz no cambiaría:

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos del proveedor de datos según la función de comparación especificada o una o varias opciones de clasificación especificadas.

Este método activa el evento `modelChanged` con el nombre de evento `sort`.

Ejemplo

En este ejemplo la ordenación se basa en las etiquetas en mayúsculas. Los elementos `a` y `b` se pasan a la función y contienen campos `label` y `data`:

```
myList.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

DataProvider.sortItemsBy()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myDP.sortItemsBy(fieldName, optionsFlag)
myDP.sortItemsBy(fieldName, order)
```

Parámetros

fieldName Cadena que especifica el nombre del campo que se va a utilizar para la ordenación. Normalmente, este valor es `"label"` o `"data"`.

order Cadena que especifica si los elementos deben clasificarse en orden ascendente (`"ASC"`) o descendente (`"DESC"`).

optionsFlag Permite realizar muchos tipos distintos de ordenaciones en una única matriz sin necesidad de duplicar toda la matriz ni de volver a ordenarla repetidas veces. Este parámetro es opcional.

Los valores posibles de *optionsFlag* son:

- `Array.DESCENDING`: ordena de mayor a menor.
- `Array.CASEINSENSITIVE`: ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`: ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas, que se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado esa etiqueta.
- `Array.UNIQUESORT`: si dos objetos de la matriz son idénticos o lo son sus campos de ordenación, este método devuelve un código de error (0) en lugar de una matriz ordenada.
- `Array.RETURNINDEXEDARRAY`: devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, la siguiente matriz devolvería la segunda línea de código y la matriz no cambiaría:

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos del proveedor de datos en el orden especificado mediante el nombre de campo especificado. Si los elementos de *`fieldName`* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *`fieldName`* suele ser `"label"` o `"data"`, pero los programadores avanzados pueden especificar cualquier valor primitivo.

Este método activa el evento `modelChanged` con el nombre de evento `sort`.

Esta es la forma más rápida para ordenar datos en un componente. También mantiene el estado de selección del componente. El método `sortItemsBy()` es rápido porque no ejecuta ningún código ActionScript mientras ordena. El método `sortItems()` necesita ejecutar una función de comparación de ActionScript y, por tanto, es más lento.

Ejemplo

En el código siguiente, los elementos de una lista se ordenan en sentido ascendente por sus etiquetas:

```
myDP.sortItemsBy("label", "ASC");
```

CAPÍTULO 13

Componente DataSet (sólo en Flash Professional)

13

El componente DataSet permite al usuario trabajar con datos como colecciones de objetos con los que se pueden realizar operaciones tales como indexar, ordenar, buscar, filtrar y modificar.

Entre las funciones del componente DataSet están DataSetIterator, un conjunto de métodos que permite atravesar y manipular una colección de datos, y DeltaPacket, un conjunto de interfaces y clases que permiten trabajar con actualizaciones de colecciones de datos. En la mayoría de los casos, dichas clases e interfaces no se utilizan directamente sino indirectamente mediante métodos establecidos por la clase DataSet.

Los elementos administrados por el componente DataSet también se denominan *objetos de transferencia*. Un objeto de transferencia muestra los datos de negocio que se encuentran en el servidor con atributos públicos o métodos de descriptores de acceso para la lectura y escritura de datos. El componente DataSet permite a los desarrolladores trabajar con objetos sofisticados de la parte del cliente que reflejan su correspondiente de la parte del servidor; o, en su forma más sencilla, una colección de objetos anónimos con atributos públicos que representan los campos de un registro de datos. Para obtener información más detallada sobre los objetos de transferencia, consulte Core J2EE Patterns Transfer Object en http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html.

NOTA

El componente DataSet necesita Flash Player 7 o una versión posterior.

Utilización del componente DataSet

Normalmente, se utiliza el componente DataSet en combinación con otros componentes para manipular y actualizar un origen de datos: un componente conector que permite realizar una conexión con un origen de datos externo, componentes de interfaz de usuario que permiten visualizar datos del origen de datos y un componente Resolver que permite convertir actualizaciones realizadas en el juego de datos en un formato adecuado para enviarlo al origen de datos externo. A continuación podrá utilizarse la vinculación de datos para vincular entre sí las propiedades de los distintos componentes.

El componente DataSet utiliza funcionalidad en las clases de vinculación de datos. Si se desea trabajar con el componente DataSet únicamente en ActionScript, sin utilizar las fichas Vinculaciones y Esquema del inspector de componentes para establecer propiedades, será necesario importar las clases de vinculación de datos en el archivo FLA y establecer las propiedades necesarias en el código. Véase "Disponibilidad de las clases de vinculación de datos en tiempo de ejecución (sólo en Flash Professional)" en la página 217.

Para obtener información general sobre cómo administrar datos en Flash mediante el componente DataSet, consulte "Administración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Parámetros de DataSet

Se pueden establecer los siguientes parámetros para el componente DataSet:

itemClassName es una cadena que indica el nombre de la clase del objeto de transferencia para el que se genera una instancia cada vez que se crea un elemento nuevo en el componente DataSet.

El componente DataSet utiliza objetos de transferencia para representar los datos que se recuperan de un origen de datos externo. Si deja este parámetro en blanco, el juego de datos crea un objeto de transferencia anónimo. Si le asigna un valor a este parámetro, el juego de datos crea una instancia para el objeto de transferencia cuando se añaden nuevos datos.

NOTA	Debe realizar una referencia completa a esta clase en algún lugar del código para asegurarse de que se compila en la aplicación (como `private var myItem:my.package.myItem;`).

logChanges es un valor booleano que se establece de forma predeterminada como `true`. Si este parámetro se establece como `true`, el juego de datos registra todos los cambios realizados en sus datos y cualquier llamada de método efectuada en los objetos de transferencia asociados.

readOnly es un valor booleano que se establece de forma predeterminada como `false`. Si se establece este parámetro como `true`, el juego de datos no puede modificarse.

Puede escribir código ActionScript para utilizar las propiedades, métodos y eventos del componente DataSet para controlar estas opciones y otras adicionales. Para más información, consulte "Clase DataSet (sólo en Flash Professional)" en la página 352.

Flujo de trabajo normal del componente DataSet

El flujo de trabajo normal del componente DataSet es el que se muestra a continuación.

Para utilizar el componente DataSet:

1. Añada una instancia del componente DataSet a su aplicación y asígnele un nombre de instancia.
2. Seleccione la ficha Esquema para el componente DataSet y cree propiedades de componente para representar los campos continuos del juego de datos.
3. Cargue el componente DataSet con datos de un origen de datos externo. (Para más información, consulte "Carga de datos en el componente DataSet" en *Utilización de Flash*.)
4. Utilice la ficha Vinculaciones del inspector de componentes para vincular los campos del juego de datos a componentes de la interfaz de usuario de su aplicación.

   Los controles de interfaz de usuario se notifican cuando se seleccionan o modifican registros (objetos de transferencia) en el componente DataSet y se actualizan como corresponde. Además, el componente DataSet recibe notificación de los cambios realizados en el control de la interfaz de usuario; el juego de datos lleva a cabo un seguimiento de dichos cambios, que pueden extraerse con un paquete delta.
5. Llame a los métodos del componente DataSet en su aplicación para administrar los datos.

NOTA	Además de estos pasos, también puede vincular el componente DataSet a un componente Connector o Resolver para obtener una solución completa para acceder, administrar y actualizar datos de un origen de datos externo.

Creación de aplicaciones con el componente DataSet

Normalmente se utiliza el componente DataSet con otros componentes de interfaz de usuario y, a menudo, con un componente conector como XMLConnector o WebServiceConnector. Los elementos del juego de datos se llenan mediante el componente conector o datos ActionScript sin formato y, a continuación, se vinculan con controles de interfaz de usuario (como los componentes List o DataGrid).

El componente DataSet utiliza funcionalidad en las clases de vinculación de datos. Si se desea trabajar con el componente DataSet únicamente en ActionScript, sin utilizar las fichas Vinculaciones y Esquema del inspector de componentes para establecer propiedades, será necesario importar las clases de vinculación de datos en el archivo FLA y establecer las propiedades necesarias en el código. Véase "Disponibilidad de las clases de vinculación de datos en tiempo de ejecución (sólo en Flash Professional)" en la página 217.

Para crear una aplicación con el componente DataSet:

1. En Flash Professional 8, seleccione Archivo > Nuevo. En la columna Tipo, seleccione Documento de Flash y haga clic en Aceptar.
2. Abra el panel Componentes si aún no está abierto.
3. Arrastre un componente DataSet desde el panel Componentes al escenario. En el inspector de propiedades, asígnele el nombre de instancia **user_ds**.
4. Arrastre un componente DataGrid al escenario y asígnele el nombre de instancia **user_dg**.
5. Cambie el tamaño del componente DataGrid para que tenga aproximadamente 300 píxeles de ancho y 100 de alto.
6. Arrastre un componente Button al escenario y asígnele el nombre de instancia **next_button**.
7. En la línea de tiempo, seleccione el primer fotograma de la capa 1 y abra el panel Acciones.
8. Añada el código siguiente al panel Acciones:

```
var recData_array:Array = [{id:0, firstName:"Mick", lastName:"Jones"},
              {id:1, firstName:"Joe", lastName:"Strummer"},
              {id:2, firstName:"Paul", lastName:"Simonon"}];
user_ds.items = recData_array;
```

   De esta forma se llena la propiedad `items` del objeto DataSet con una matriz de objetos que tienen tres propiedades cada uno: `id`, `firstName` y `lastName`.

9. Añada las tres propiedades y sus tipos de datos necesarios al esquema DataSet:
   a. Seleccione el componente DataSet del escenario, abra el inspector de componentes y haga clic en la ficha Esquema.
   b. Haga clic en Añadir una propiedad de componente y añada tres propiedades nuevas, con los nombres de campo `id`, `firstName` y `lastName`, y los tipos de datos `Number`, `String` y `String`, respectivamente.

   Si prefiere añadir al código las propiedades y sus tipos de datos necesarios, puede añadir la siguiente línea de código en el panel Acciones, en lugar de los pasos a y b anteriormente mencionados:

```
// Añadir los tipos de esquema necesarios.
var i:mx.data.types.Str;
var j:mx.data.types.Num;
```

10. Para vincular el contenido del componente DataSet con el del componente DataGrid, abra el inspector de componentes y haga clic en la ficha Vinculaciones.
11. Seleccione el componente DataGrid (`user_dg`) en el escenario y haga clic en el botón Añadir vinculación (+) del inspector de componentes.

12. En el cuadro de diálogo Añadir vinculación, seleccione “dataProvider : Array” y haga clic en Aceptar.
13. Haga doble clic en el campo Vinculado a del inspector de componentes.
14. En el cuadro de diálogo Vinculado a que aparece, seleccione “DataSet <user_ds>” en la columna Ruta del componente y, a continuación, seleccione “dataProvider : Array” en la columna Ubicación del esquema.
15. Para vincular el índice seleccionado del componente DataSet al índice seleccionado del componente DataGrid, seleccione el componente DataGrid en el escenario y haga clic nuevamente en el botón Añadir vinculación (+) del inspector de componentes.
16. En el cuadro de diálogo que aparece, seleccione “selectedIndex : Number”. Haga clic en Aceptar.
17. Haga doble clic en el campo Vinculado a del inspector de componentes para abrir el cuadro de diálogo Vinculado a.
18. En el campo Ruta del componente, seleccione “DataSet <user_ds>” en la columna Ruta del componente y, a continuación, seleccione “selectedIndex : Number” en la columna Ubicación del esquema.
19. Introduzca el código siguiente en el panel Acciones:

```
next_button.addEventListener("click", nextBtnClick);
function nextBtnClick(evt_obj:Object):Void {
  user_ds.next();
}
```

    En este código se utiliza el método `DataSet.next()` para desplazarse al siguiente elemento de la colección de elementos del objeto DataSet. Como ya se había vinculado con anterioridad la propiedad `selectedIndex` del objeto DataGrid a la misma propiedad del objeto DataSet, si se cambia el elemento actual del objeto DataSet, también cambiará el elemento actual (seleccionado) del objeto DataGrid.
20. Guarde el archivo y seleccione Control > Probar película para probar el archivo SWF.

    El objeto DataGrid se llenará con los elementos especificados. Fíjese en cómo al hacer clic en el botón cambia el elemento seleccionado en el objeto DataGrid.

Clase DataSet (sólo en Flash Professional)

Herencia MovieClip > DataSet

Nombre de clase de ActionScript mx.data.components.DataSet

El componente DataSet permite al usuario trabajar con datos como colecciones de objetos con los que se pueden realizar operaciones tales como indexar, ordenar, buscar, filtrar y modificar.

Entre las funciones del componente DataSet están DataSetIterator, un conjunto de métodos que permite atravesar y manipular una colección de datos, y DeltaPacket, un conjunto de interfaces y clases que permiten trabajar con actualizaciones de colecciones de datos. En la mayoría de los casos, dichas clases e interfaces no se utilizan directamente sino indirectamente mediante métodos establecidos por la clase DataSet.

Resumen de métodos de la clase DataSet

En la tabla siguiente se enumeran los métodos de la clase DataSet.

Método	Descripción
`DataSet.addItem()`	Añade el elemento especificado a la colección.
`DataSet.addItemAt()`	Añade un elemento al juego de datos en la posición especificada.
`DataSet.addSort()`	Crea una nueva vista ordenada de los elementos de la colección.
`DataSet.applyUpdates()`	Emite una señal que indica que la propiedad `deltaPacket` tiene un valor al que se puede acceder mediante la vinculación de datos o utilizando ActionScript.
`DataSet.changesPending()`	Indica si la colección tiene cambios pendientes que no se hayan enviado aún a un paquete delta.
`DataSet.clear()`	Elimina todos los elementos de la vista actual de la colección.
`DataSet.createItem()`	Devuelve un elemento de la colección que acaba de inicializarse.
`DataSet.disableEvents()`	Detiene el envío de eventos DataSet a los detectores.
`DataSet.enableEvents()`	Reanuda el envío de eventos DataSet a los detectores.
`DataSet.find()`	Busca un elemento en la vista actual de la colección.
`DataSet.findFirst()`	Busca la primera vez que aparece un elemento en la vista actual de la colección.

Método	Descripción
DataSet.findLast()	Busca la última vez que aparece un elemento en la vista actual de la colección.
DataSet.first()	Se desplaza al primer elemento de la vista actual de la colección.
DataSet.getItemId()	Devuelve el ID exclusivo del elemento especificado.
DataSet.getIterator()	Devuelve un clon del repetidor actual.
DataSet.getLength()	Devuelve el número de elementos del juego de datos.
DataSet.hasNext()	Indica si el repetidor actual se encuentra o no al final de la vista correspondiente de la colección.
DataSet.hasPrevious()	Indica si el repetidor actual se encuentra o no al principio de la vista correspondiente de la colección.
DataSet.hasSort()	Indica si la ordenación especificada existe o no.
DataSet.isEmpty()	Indica si la colección contiene elementos o no.
DataSet.last()	Se desplaza al último elemento de la vista actual de la colección.
DataSet.loadFromSharedObj()	Carga todos los datos necesarios para restaurar esta colección DataSet a partir de un objeto compartido.
DataSet.locateById()	Desplaza el repetidor actual al elemento que tenga el ID especificado.
DataSet.next()	Se desplaza al siguiente elemento de la vista actual de la colección.
DataSet.previous()	Se desplaza al elemento anterior de la vista actual de la colección.
DataSet.removeAll()	Elimina todos los elementos de la colección.
DataSet.removeItem()	Elimina el elemento especificado de la colección.
DataSet.removeItemAt()	Elimina un elemento de juego de datos en una posición especificada.
DataSet.removeRange()	Elimina la configuración del rango del repetidor actual.
DataSet.removeSort()	Elimina el orden especificado del objeto DataSet.
DataSet.saveToSharedObj()	Guarda los datos del objeto DataSet en un objeto compartido.
DataSet.setIterator()	Establece el repetidor actual del objeto DataSet.
DataSet.setRange()	Establece la configuración del rango del repetidor actual.

Método	Descripción
DataSet.skip()	Se desplaza hacia adelante o hacia atrás un número especificado de elementos en la vista actual de la colección.
DataSet.useSort()	Convierte en ordenación activa la que se haya especificado.

Resumen de propiedades de la clase DataSet

En la siguiente tabla se enumeran las propiedades de la clase DataSet.

Propiedad	Descripción
DataSet.currentItem	Devuelve el elemento actual de la colección.
DataSet.dataProvider	Devuelve el proveedor de datos.
DataSet.deltaPacket	Devuelve los cambios realizados en la colección o asigna los cambios que se realizarán en la colección.
DataSet.filtered	Indica si los elementos están filtrados o no.
DataSet.filterFunc	Función definida por el usuario para filtrar elementos de la colección.
DataSet.items	Elementos de la colección.
DataSet.itemClassName	Nombre del objeto que va a crearse al asignar elementos.
DataSet.length	Especifica el número de elementos de la vista actual de la colección.
DataSet.logChanges	Indica si se registran o no los cambios realizados en la colección o en sus elementos.
DataSet.properties	Contiene las propiedades (campos) de los objetos de transferencia que existen en esta colección.
DataSet.readOnly	Indica si la colección puede modificarse o no.
DataSet.schema	Especifica el esquema de la colección en formato XML.
DataSet.selectedIndex	Contiene el índice del elemento actual en la colección.

Resumen de eventos de la clase DataSet

En la tabla siguiente se enumeran los eventos de la clase DataSet.

Evento	Descripción
DataSet.addItem	Se difunde antes de añadir un elemento a la colección.
DataSet.afterLoaded	Se difunde después de asignar la propiedad `items`.

Evento	Descripción
DataSet.calcFields	Se difunde cuando deben actualizarse los campos calculados.
DataSet.deltaPacketChanged	Se difunde cuando el paquete delta del objeto DataSet ha sufrido alteraciones y está preparado para utilizarse.
DataSet.iteratorScrolled	Se difunde cuando cambia la posición del repetidor.
DataSet.modelChanged	Se difunde cuando se ha realizado algún cambio en los elementos de la colección.
DataSet.newItem	Se difunde cuando el objeto DataSet genera un nuevo objeto de transferencia, pero antes de añadirlo a la colección.
DataSet.removeItem	Se difunde antes de eliminar un elemento.
DataSet.resolveDelta	Se difunde cuando se asigna un paquete delta al objeto DataSet que contiene mensajes.

DataSet.addItem

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.addItem = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("addItem", listenerObject);
```

Sintaxis 2:

```
on (addItem) {
  // ...
}
```

Descripción

Evento; se genera justo antes de insertar un nuevo registro (objeto de transferencia) en esta colección.

Si se especifica el valor `false` en la propiedad `result` del objeto en el que se ha producido el evento, se cancelará la operación de adición; en cambio si se establece el valor `true`, podrá realizarse dicha operación.

El objeto de evento (*`eventObj`*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"addItem"`.

`item` Referencia al elemento de la colección que se va a añadir.

`result` Valor booleano que especifica si el elemento especificado debe añadirse o no. De forma predeterminada, este valor es `true`.

Ejemplo

El controlador de eventos `addItem` siguiente cancela la adición del nuevo elemento si una función definida por el usuario denominada `userHasAdminPrivs()` devuelve el valor `false`; de lo contrario, sí se admite la adición.

```
function userHasAdminPrivs():Boolean {
  return false; // Cambiar esto a true para permitir inserciones.
}

my_ds.addEventListener("addItem", addItemListener);
my_ds.addItem({name:"Bobo", occupation:"clown"});

function addItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Permitir la adición del elemento.
    evt_obj.result = true;
    trace("Item added");
  } else {
    // No permitir la adición del elemento; el usuario no tiene privilegios
  de administrador.
    evt_obj.result = false;
    trace("Error, insufficient permissions");
  }
}
```

Véase también

`DataSet.removeItem`

DataSet.addItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.addItem([*obj*])

Parámetros

obj Objeto que se añadirá a esta colección. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se añadió a la colección, `false` si no se añadió.

Descripción

Método; añade a la colección el registro (objeto de transferencia) especificado para que pueda administrarse. El elemento recién añadido se convierte en el elemento actual del juego de datos. Si no se especifica ningún parámetro *obj*, se crea automáticamente un nuevo objeto mediante `DataSet.createItem()`.

La ubicación del nuevo elemento en la colección depende de si se ha especificado o no una ordenación para el repetidor actual. Si no se está utilizando ninguna ordenación, el elemento se añade al final de la colección. Si se está utilizando una ordenación, el elemento se añade a la colección en función de la posición que ocupe en la ordenación actual.

Para más información sobre la inicialización y generación de objetos de transferencia, consulte `DataSet.createItem()`.

Ejemplo

En el siguiente ejemplo se utiliza `DataSet.addItem()` para generar un elemento nuevo y añadirlo al juego de datos:

```
my_ds.addEventListener("addItem", addItemListener);
my_ds.addItem({name:"Bobo", occupation:"clown"});

function addItemListener(evt_obj:Object):Void {
  trace("adding item");
}
```

En el siguiente ejemplo se demuestra cómo se puede aceptar o rechazar una inserción de elemento en DataSet estableciendo el resultado en `true` o en `false` en el controlador del evento `addItem`. Arrastre un componente DataSet al escenario y asígnele el nombre de instancia `my_ds`. Arrastre un componente DataGrid al escenario y asígnele el nombre de instancia `my_dg`. Arrastre un componente CheckBox al escenario y asígnele el nombre de instancia `my_ch`. Arrastre un componente Button al escenario y asígnele el nombre de instancia `submit_button`. Añada dos propiedades, `name` y `occupation`, al componente DataSet mediante la ficha Esquema del inspector de componentes. Cree una vinculación entre la propiedad `my_ds.dataProvider` y la propiedad `my_dg.dataProvider` mediante el panel Vinculaciones del inspector de componentes. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
my_ds.addEventListener("addItem", addItemListener);
submit_button.addEventListener("click", submitListener);
function userHasAdminPrivs():Boolean {
  return my_ch.selected;
}
function addItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Permitir la adición del elemento.
    evt_obj.result = true;
    trace("Item added");
  } else {
    // No permitir la adición del elemento; el usuario no tiene privilegios
  de administrador.
    evt_obj.result = false;
    trace("Error, insufficient permissions");
  }
}
function submitListener(evt_obj:Object):Void {
  my_ds.addItem({name:"bobo", occupation:"clown"});
}
```

Véase también

`DataSet.createItem()`

DataSet.addItemAt()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

dataSetInstance.addItemAt(*index*, *item*)

Parámetros

index Número igual o mayor que 0. Este número indica la posición en la que se debe insertar el elemento; es el índice del nuevo elemento.

item Objeto que contiene los datos del elemento.

Valor devuelto

Un valor booleano que indica si el elemento se ha añadido o no: `true` indica que el elemento se ha añadido y `false` indica que el elemento ya existe en el juego de datos.

Descripción

Método; añade un nuevo elemento al juego de datos en el índice especificado. Se omiten los índices cuya longitud sea mayor que la del proveedor de datos.

Este método activa el evento `modelChanged` con el tipo de evento `addItem`.

Ejemplo

En el siguiente ejemplo se utiliza el método `addItemAt()` para añadir un elemento al componente DataSet en la primera posición:

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});
my_ds.addItemAt(0, {name:"Bobo", years:1});
```

DataSet.addSort()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.addSort(*name*, *fieldList*, *sortOptions*)

Parámetros

name Cadena que especifica el nombre de la ordenación.

fieldList Matriz de cadenas que especifica los nombres de campo en los que basar la ordenación.

sortOptions Uno o varios de los valores enteros (constantes) siguientes, que indican las opciones que se utilizan en esta ordenación. Separe los distintos valores mediante el operador OR en modo bit (|). Especifique uno o varios de los siguientes valores:

- `DataSetIterator.Ascending` Ordena los elementos en sentido ascendente. Esta es la opción de ordenación predeterminada si no se especifica ninguna.
- `DataSetIterator.Descending` Ordena los elementos en sentido descendente en función de las propiedades de elemento especificadas.
- `DataSetIterator.Unique` Impide la ordenación si existen campos con valores idénticos.
- `DataSetIterator.CaseInsensitive` No distingue entre mayúsculas y minúsculas al comparar dos cadenas durante la ordenación. De forma predeterminada, la ordenación distingue entre mayúsculas y minúsculas cuando la propiedad en la que se basa es una cadena.

Se emite una excepción `DataSetError` cuando se especifica `DataSetIterator.Unique` como opción de ordenación y los datos que se van a ordenar no son exclusivos, cuando ya se ha añadido el nombre de ordenación especificado o cuando la propiedad especificada en la matriz *fieldList* no existe en este juego de datos.

Valor devuelto

Ninguno.

Descripción

Método; crea un nuevo orden ascendente o descendente para el repetidor actual en función de las propiedades especificadas en el parámetro *fieldList*. Flash asigna la nueva ordenación automáticamente al repetidor actual tras su creación y, a continuación, la guarda en la colección de ordenaciones para poder recuperarla en otro momento.

Ejemplo

En el código siguiente se crea una nueva ordenación denominada "nameSort" que realiza una ordenación en sentido descendente y que no distingue entre mayúsculas y minúsculas en el campo "name" del objeto DataSet.

```
import mx.data.components.datasetclasses.DataSetIterator;

my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addSort("nameSort", ["name"], DataSetIterator.Descending |
  DataSetIterator.Unique | DataSetIterator.CaseInsensitive);
```

En el siguiente ejemplo, se pueden añadir datos de forma dinámica al componente DataSet introduciendo el nombre y apellido en las instancias del componente TextInput del escenario y haciendo clic en el botón Enviar. Tras haber añadido elementos al componente DataSet, se puede borrar el juego de datos haciendo clic en el botón Borrar del escenario. Arrastre un componente DataGrid al escenario y asígnele el nombre de instancia `my_dg`. Arrastre dos instancias del componente Button al escenario y asígneles los nombres de instancia `submit_button` y `clear_button`. Arrastre un componente DataSet al escenario y asígnele el nombre de instancia `my_ds`. Arrastre dos componentes TextInput al escenario y asígneles los nombres de instancia `firstName_ti` y `lastName_ti`. Arrastre un componente Alert a la biblioteca del documento actual. Añada dos propiedades de componente, `firstName` y `lastName`, al esquema del componente DataSet mediante la ficha Esquema del inspector de componentes. A continuación, añada una vinculación de datos desde la propiedad `dataProvider` del componente DataSet hasta la propiedad `dataProvider` del componente DataGrid mediante la ficha Vinculaciones del inspector de componentes. Por último, pegue el siguiente código al primer fotograma de la línea de tiempo principal:

```
import mx.controls.Alert;

my_ds.addSort("lastFirst", ["lastName", "firstName"]);

my_dg.enabled = false;
clear_button.enabled = false;
submit_button.label = "Submit";
clear_button.label = "Clear";
```

```
my_ds.addEventListener("addItem", addItemListener);
my_ds.addEventListener("modelChanged", modelChangedListener);
submit_button.addEventListener("click", submitListener);
clear_button.addEventListener("click", clearListener);

function modelChangedListener(evt_obj:Object):Void {
  my_dg.enabled = (evt_obj.target.length > 0);
  clear_button.enabled = my_dg.enabled;
}
function submitListener(evt_obj:Object):Void {
  my_ds.addItem({firstName:firstName_ti.text, lastName:lastName_ti.text});
}
function addItemListener(evt_obj:Object):Void {
  if ((evt_obj.item.firstName.length == 0) || (evt_obj.item.lastName.length
  == 0)) {
    Alert.show("Error, first name or last name cannot be blank.", "Error",
  Alert.OK, _level0);
    evt_obj.result = false;
  } else {
    firstName_ti.text = "";
    lastName_ti.text = "";
  }
}
function clearListener(evt_obj:Object):Void {
  Alert.show("Are you sure you want to clear the data?", "Warning",
  Alert.OK | Alert.CANCEL, _level0, clearConfirmListener);
}
function clearConfirmListener(evt_obj:Object):Void {
  switch (evt_obj.detail) {
  case Alert.OK:
    my_ds.clear();
    break;
  case Alert.CANCEL:
    break;
  }
}
```

Seleccione Control > Probar película para probar el documento en el entorno de edición. Introduzca algún texto en ambas instancias de TextInput y haga clic en el botón Enviar. Debería añadirse un nuevo elemento a la instancia de DataGrid. Al hacer clic en el botón Borrar, debería aparecer una instancia del componente Alert con un mensaje de confirmación donde se le pregunte si desea borrar el contenido de DataGrid. Si hace clic en Aceptar, se borrará la propiedad `dataProvider` del componente DataSet (y luego la propiedad dataProvider del componente DataGrid, debido a la vinculación). Si hace clic en Cancelar, se descarta la instancia del componente Alert.

Véase también

`DataSet.removeSort()`

DataSet.afterLoaded

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.afterLoaded = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("afterLoaded", listenerObject);
```

Sintaxis 2:

```
on (afterLoaded) {
  // ...
}
```

Descripción

Evento; se difunde inmediatamente después de asignar la propiedad `DataSet.items`.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"afterLoaded"`.

Ejemplo

En el siguiente ejemplo, aparece un formulario denominado `contactForm` (que no se muestra) cuando ya se han asignado los elementos del juego de datos `contact_ds`.

```
contact_ds.addEventListener("afterLoaded", loadListener);
var loadListener:Object = new Object();
loadListener.afterLoaded = function (evt_obj:Object) {
  if (evt_obj.target == "contact_ds") {
    contactForm.visible = true;
  }
};
```

En el siguiente ejemplo se utiliza el evento `afterLoaded` del componente DataSet para llenar la propiedad `dataProvider` de un componente List en el escenario. Arrastre un componente List y un componente DataSet al escenario y asígneles los nombres de instancia `my_list` y `my_ds` respectivamente. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
my_list.labelField = "name";

var itemsListener:Object = new Object();
itemsListener.afterLoaded = function (evt_obj:Object):Void {
  trace("After loaded");
  my_list.dataProvider = evt_obj.target.items;
}
my_ds.addEventListener("afterLoaded", itemsListener);

var item_array:Array = [{name:"Douglas"}, {name:"Vinnie"},
  {name:"Katherine"}, {name:"David"}];
my_ds.items = item_array;
```

DataSet.applyUpdates()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance`.applyUpdates()`

Valor devuelto

Ninguno.

Descripción

Método; emite una señal que indica que la propiedad `DataSet.deltaPacket` tiene un valor al que se puede acceder mediante la vinculación de datos o directamente utilizando ActionScript. Antes de llamar a este método, el valor de la propiedad `DataSet.deltaPacket` es `null`. Este método no tiene ninguna consecuencia si se han desactivado los eventos mediante el método `DataSet.disableEvents()`.

Al llamar a este método también se crea un ID de transacción para la propiedad `DataSet.deltaPacket` actual y se emite un evento `deltaPacketChanged`. Para más información, consulte `DataSet.deltaPacket`.

Ejemplo

En el código siguiente se llama al método `applyUpdates()` en la instancia `my_ds` de DataSet.

```
my_ds.applyUpdates();
```

En el siguiente ejemplo se añaden cuatro elementos a la instancia `my_ds` de DataSet en el escenario y se muestra cada elemento en el nivel superior de la propiedad `deltaPacket`:

```
my_ds.addItem({name:"Thomas", age:35, gender:"M"});
my_ds.addItem({name:"Orville", age:33, gender:"M"});
my_ds.addItem({name:"Jonathan", age:48, gender:"M"});
my_ds.addItem({name:"Carol", age:31, gender:"F"});

my_ds.applyUpdates();
var i:String;
for (i in my_ds.deltaPacket) {
  trace(i + ":\t" + my_ds.deltaPacket[i]);
}
```

Véase también

`DataSet.deltaPacket`

DataSet.calcFields

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.calcFields = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("calcFields", listenerObject);
```

Sintaxis 2:

```
on (calcFields) {
  // ...
}
```

Descripción

Evento; se genera cuando deben determinarse los valores de los campos calculados del elemento actual de la colección. Un campo calculado es un campo que tiene la propiedad Kind establecida en Calculated en la ficha Esquema del inspector de componentes. El detector de eventos `calcFields` creado por el usuario debe realizar el cálculo necesario y establecer el valor correspondiente al campo calculado.

También se llama a este evento cuando se actualiza el valor de un campo no calculado (es decir, un campo cuya propiedad Kind está establecida en Data en la ficha Esquema del inspector de componentes).

Para más información sobre la propiedad Kind, consulte "Tipos de esquema" en *Utilización de Flash*.

ATENCIÓN	No cambie ninguno de los valores de los campos no calculados de este evento, ya que ello provocaría un "bucle infinito". Establezca únicamente los valores de los campos calculados dentro del evento `calcFields`.

DataSet.changesPending()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.changesPending()`

Valor devuelto

Valor booleano.

Descripción

Método; devuelve `true` si la colección, o cualquier elemento de la colección, tiene cambios pendientes de enviar en un paquete delta; de lo contrario, devuelve `false`.

Ejemplo

En el código siguiente se activa el botón Guardar cambios (que no se muestra) si la colección DataSet, o cualquier elemento de la colección, ha sufrido modificaciones no confirmadas todavía en un paquete delta.

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addEventListener("modelChanged", modelChangedListener);
function modelChangedListener(evt_obj:Object):Void {
  if (evt_obj.target.changesPending()) {
    trace("changes pending");
    submitChanges_button.enabled = true;
  }
}
submitChanges_button.enabled = false;
my_ds.addItem({name:"Hal", years:4});
```

DataSet.clear()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.clear()

Valor devuelto

Ninguno.

Descripción

Método; elimina los elementos de la vista actual de la colección. La consideración de un elemento como "apto para ser visto" depende de la configuración de filtro y de rango actual especificada en el repetidor actual. Por tanto, es posible que la llamada a este método no borre la totalidad de los elementos de la colección. Si se desea borrar la totalidad de los elementos de una colección, con independencia de la vista del repetidor actual, utilice `DataSet.removeAll()`.

Si el valor de `DataSet.logChanges` es `true` al invocar este método, las entradas eliminadas se añaden a `DataSet.deltaPacket` para todos los elementos que se encuentran dentro de la colección.

Ejemplo

En el siguiente ejemplo se eliminan todos los elementos de la vista actual de la colección DataSet. La eliminación de los elementos se registrará porque la propiedad `logChanges` tiene establecido el valor `true`.

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addSort("nameSort", ["name"]);
my_ds.filtered = true;
my_ds.filterFunc = function(item:Object):Boolean {
  return (item.years >= 3);
};
my_ds.logChanges = true;
my_ds.clear(); // Eliminar elementos filtrados del juego de datos.
my_ds.removeSort("nameSort");
```

Véase también

`DataSet.deltaPacket`, `DataSet.logChanges`

DataSet.createItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.createItem([`*`itemData`*`])`

Parámetros

`itemData` Datos asociados con el elemento. Este parámetro es opcional.

Valor devuelto

El elemento que acaba de generarse.

Descripción

Método; genera un elemento que no está asociado con la colección. Es posible especificar la clase de objeto creada con la propiedad `DataSet.itemClassName`. Si no se especifica un valor `DataSet.itemClassName` y se omite el parámetro *`itemData`*, se generará un objeto anónimo. Las propiedades de este objeto anónimo tienen los valores predeterminados en función del esquema actual que se haya especificado en `DataSet.schema`.

Al invocar este método se envía una notificación a los detectores del evento `DataSet.newItem`, que pueden entonces manipular el elemento antes de que el método lo devuelva. Los datos opcionales del elemento se utilizan para inicializar la clase especificada con la propiedad `DataSet.itemClassName` o se utilizan como elemento si `DataSet.itemClassName` está en blanco.

Se emite una excepción `DataSetError` cuando la clase especificada con la propiedad `DataSet.itemClassName` no puede cargarse.

Ejemplo

```
my_ds.addEventListener("newItem", newItemListener);
function newItemListener(evt_obj:Object):Void {
  trace("new item was added: {name:'" + evt_obj.item.name + "', years:" +
  evt_obj.item.years + "}");
}

my_ds.addItem(my_ds.createItem({name:"Wilson", years:3}));
my_ds.addItem({name:"Tom", years:2});

my_ds.filtered = true;
my_ds.filterFunc = function(item:Object):Boolean {
  return (item.years % 2 == 0);
};
```

Véase también

`DataSet.itemClassName`, `DataSet.newItem`, `DataSet.schema`

DataSet.currentItem

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.currentItem`

Descripción

Propiedad (sólo lectura); devuelve el elemento actual de la colección DataSet, o `null` si está vacía la colección o la vista del repetidor actual de la colección.

Esta propiedad permite el acceso directo al elemento de la colección. No se realiza ningún seguimiento de los cambios realizados a causa del acceso directo a este objeto (en la propiedad `DataSet.deltaPacket`), ni tampoco de la configuración del esquema que se haya aplicado a las propiedades de este objeto.

Ejemplo

En el ejemplo siguiente se muestra el valor de la propiedad `name` definido en el elemento actual del juego de datos denominado `customers_ds`.

```
customers_ds.addItem({name:"Milton", years:3});
customers_ds.addItem({name:"Mark", years:3});
customers_ds.addItem({name:"Sarah", years:1});
customers_ds.addItem({name:"Michael", years:2});
customers_ds.addItem({name:"Frank", years:2});

trace(customers_ds.currentItem.name); // Frank
```

DataSet.dataProvider

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.dataProvider`

Descripción

Propiedad; proveedor de datos para este juego de datos. Esta propiedad proporciona datos a los controles de interfaz de usuario, como los componentes List y DataGrid.

Para más información sobre la interfaz API de DataProvider, consulte "Interfaz API de DataProvider" en la página 333.

Ejemplo

En el código siguiente se asigna la propiedad `dataProvider` de un objeto DataSet a la propiedad correspondiente de un componente DataGrid.

```
my_dg.dataProvider = my_ds.dataProvider;
```

DataSet.deltaPacket

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.deltaPacket

Descripción

Propiedad; devuelve un paquete delta que contiene todas las operaciones de cambio realizadas en la colección *dataSet* y en sus elementos. El valor de esta propiedad es `null` hasta que se llama a `DataSet.applyUpdates()` en *dataSet*.

Cuando se llama a `DataSet.applyUpdates()`, se asigna un ID de transacción al paquete delta. Este ID de transacción se utiliza para identificar el paquete delta en un proceso de actualización de ida y vuelta entre el servidor y el cliente. Cualquier asignación posterior que realice el paquete delta a la propiedad `deltaPacket` con un ID de transacción coincidente tiene la función de respuesta del servidor a los cambios enviados con anterioridad. Se utiliza un paquete delta con un ID coincidente para actualizar la colección e informar de los errores especificados dentro del paquete.

Los errores o mensajes del servidor se notifican a los detectores del evento `DataSet.resolveDelta`. Observe que se omite la configuración de `DataSet.logChanges` cuando se asigna un paquete delta con ID coincidente a `DataSet.deltaPacket`. Si un paquete delta no tiene un ID de transacción coincidente se actualiza la colección, como si la interfaz API de DataSet se utilizara directamente. Es posible que se creen entonces más entradas delta, lo cual dependerá de la configuración actual de `DataSet.logChanges` de *dataSet* y del paquete delta.

Se emitirá una excepción `DataSetError` si se asigna un paquete delta con un ID de transacción coincidente y si no se encuentra uno de los elementos recién asignados del paquete delta en el mismo.

Véase también

`DataSet.applyUpdates()`, `DataSet.logChanges`, `DataSet.resolveDelta`

DataSet.deltaPacketChanged

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.deltaPacketChanged = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("deltaPacketChanged", listenerObject);
```

Sintaxis 2:

```
on (deltaPacketChanged) {
  // ...
}
```

Descripción

Evento; se difunde cuando la propiedad `deltaPacket` del objeto DataSet especificado ha sufrido cambios y está preparada para utilizarse.

Véase también

`DataSet.deltaPacket`

DataSet.disableEvents()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

```
dataSetInstance.disableEvents()
```

Valor devuelto

Ninguno.

Descripción

Método; desactiva eventos para el objeto DataSet. Mientras los eventos están desactivados no se actualiza ningún control de interfaz de usuario (como el componente DataGrid) cuando se realizan cambios en los elementos de la colección o se desplaza el objeto DataSet hasta otro elemento de la colección.

Para volver a activar los eventos, es necesario llamar a `DataSet.enableEvents()`. El método `disableEvents()` puede llamarse varias veces y, para volver a activar la distribución de eventos, es necesario llamar al método `enableEvents()` el mismo número de veces.

Ejemplo

En el siguiente ejemplo, los eventos se desactivan antes de que los elementos de la colección sufran cambios, de forma que el rendimiento del sistema no se vea perjudicado porque el objeto DataSet intente actualizar los controles:

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Desactivar eventos para el juego de datos.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Todo al 50%
   my_ds.previous();
}

trace("After:");
traceItems();

// Indicar al juego de datos que ahora hay que actualizar los controles.
my_ds.enableEvents();

function traceItems():Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Véase también

`DataSet.enableEvents()`

DataSet.enableEvents()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.enableEvents()

Valor devuelto

Ninguno.

Descripción

Método; reactiva los eventos de los objetos DataSet después de que dichos eventos se hayan desactivado mediante una llamada a DataSet.disableEvents(). Para reactivar eventos para el objeto DataSet, debe llamarse al método enableEvents() un número de veces igual o mayor que el número de veces que se haya llamado a disableEvents().

Ejemplo

En el siguiente ejemplo, los eventos se desactivan antes de que los elementos de la colección sufran cambios, de forma que el rendimiento del sistema no se vea perjudicado porque el objeto DataSet intente actualizar los controles.

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Desactivar eventos para el juego de datos.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Todo al 50%
   my_ds.previous();
}

trace("After:");
traceItems();
```

```
// Indicar al juego de datos que ahora hay que actualizar los controles.
my_ds.enableEvents();

function traceItems():Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Véase también

`DataSet.disableEvents()`

DataSet.filtered

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.filtered`

Descripción

Propiedad; valor booleano que indica si los datos del repetidor actual están filtrados o no. El valor predeterminado es `false`. Cuando esta propiedad es `true`, se llama a la función de filtro especificada por `DataSet.filterFunc` para todos los elementos de la colección.

Ejemplo

En el ejemplo siguiente se ha activado el filtrado en el objeto DataSet denominado `employee_ds`. Supongamos que cada registro de la colección DataSet contiene un campo denominado `empType`. La función de filtro siguiente devuelve `true` si el campo `empType` del elemento actual se ha establecido en `"management"`; de lo contrario, devuelve `false`.

```
employee_ds.filtered = true;
employee_ds.filterFunc = function (item:Object) {
  // Filtrar a los empleados que ocupan cargos directivos.
  return(item.empType != "management");
};
```

En el ejemplo siguiente se llena un componente DataGrid con contenido cargado dinámicamente mediante el componente XMLConnector. Cuando un usuario hace clic en una instancia de CheckBox en el escenario, el contenido del componente DataSet se filtra y se actualiza automáticamente en el componente DataGrid.

Arrastre los siguientes componentes al escenario y asígneles los siguientes nombres de instancia:

- CheckBox (`editorsChoice_ch`)
- DataGrid (`reviews_dg`)
- DataSet (`reviews_ds`)
- XMLConnector (`reviews_xmlconn`)

Descargue una copia del siguiente documento XML y guárdelo en su disco duro local:`http://www.helpexamples.com/flash/xml/reviews.xml`. Este documento XML se cargará dinámicamente mediante el componente XMLConnector, pero el usuario utilizará la copia local para importar el esquema de XML al componente DataSet. Seleccione la instancia de XMLConnector del escenario y seleccione la ficha Esquema del inspector de componentes. Seleccione la propiedad `results` y haga clic en el botón "Importar un esquema de un archivo XML de muestra". Seleccione el documento XML que descargó en su disco duro local y haga clic en Abrir. El esquema del documento XML debería importarse en el componente DataSet. Con la instancia de XMLConnector `reviews_xmlconn` aún seleccionada en el escenario, añada una vinculación entre la matriz `reviews_xmlconn.results.reviews.review` y la propiedad `dataProvider` de la instancia `reviews_ds` del componente DataSet. Defina la dirección de la vinculación de datos en "out" en la ficha Vinculaciones. Seleccione la instancia `reviews_ds` del componente DataSet en el escenario y añada otra vinculación para la propiedad `dataProvider`. Con la instancia `reviews_ds` seleccionada, seleccione la ficha Vinculaciones del inspector de componentes. Haga clic en el botón Añadir vinculación y añada una vinculación desde la propiedad `dataProvider` del componente DataSet a la propiedad `dataProvider` de la instancia `reviews_dg` de DataGrid.

Añada el código siguiente al fotograma 1 de la línea de tiempo principal:

```
editorsChoice_ch.label = "Editor's Choice";

reviews_xmlconn.direction = "receive";
reviews_xmlconn.multipleSimultaneousAllowed = false;
reviews_xmlconn.URL = "http://www.helpexamples.com/flash/xml/reviews.xml";
reviews_xmlconn.trigger();

reviews_dg.setSize(320, 240);
reviews_dg.addColumn("name");
reviews_dg.addColumn("rating");
reviews_dg.addColumn("reviewDate");
reviews_dg.getColumnAt(0).width = 100;
```

```
reviews_dg.getColumnAt(1).width = 100;
reviews_dg.getColumnAt(2).width = 100;
reviews_dg.setStyle("alternatingRowColors", [0xFFFFFF, 0xF6F6F6]);

editorsChoice_ch.addEventListener("click", editorsChoiceListener);
function editorsChoiceListener(evt_obj:Object):Void {
  reviews_ds.filtered = evt_obj.target.selected;
  reviews_ds.filterFunc = function(item:Object):Boolean {
    return (item.editorsChoice == 1);
  };
}
```

Seleccione Control > Probar película. De forma predeterminada, el componente DataGrid debería mostrar todas la revisiones del documento XML externo. Si hace clic en el componente CheckBox elegido por el editor, se filtrará el componente DataSet y sólo se mostrarán las revisiones específicas marcadas como elegidas por el editor.

Véase también

`DataSet.filterFunc`

DataSet.filterFunc

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

```
dataSetInstance.filterFunc = function (item:Object):Boolean {
  // devolver true|false;
};
```

Descripción

Propiedad; especifica una función que determina los elementos que se incluyen en la vista actual de la colección. Cuando `DataSet.filtered` se establece en `true`, se llama a la función asignada a esta propiedad para cada registro (objeto de transferencia) de la colección. Para cada elemento que se pase a la función, deberá devolver el valor `true` si el elemento debe incluirse en la vista actual, o `false` si no debe incluirse.

Al cambiar la función de filtro del juego de datos, es necesario establecer la propiedad `filtered` en `false` y, a continuación, en `true` de nuevo para que se genere el evento `modelChanged` apropiado. El cambio de la propiedad `filterFunc` no generará el evento.

Además, si ya hay filtro cuando se cargan los datos en (`modelChanged` o `updateAll`), éste no se aplica hasta que `filtered` se establece en `false` y de nuevo en `true`.

Ejemplo

En el ejemplo siguiente se ha activado el filtrado en el objeto DataSet denominado `employee_ds`. La función de filtro especificada devuelve `true` si el campo `empType` de cada elemento se ha establecido en `"management"`; de lo contrario, devuelve `false`.

```
employee_ds.filtered = true;
employee_ds.filterFunc = function (item:Object):Boolean {
  // Filtrar a los empleados que ocupan cargos directivos.
  return(item.empType != "management");
};
```

En el siguiente ejemplo, el usuario filtra el contenido de un componente DataSet en función del elemento seleccionado en un componente ComboBox. El componente ComboBox permite seleccionar todas las personas, sólo hombres o sólo mujeres.

Arrastre un componente DataSet, DataGrid y ComboBox al escenario y asígneles los nombres de instancia `my_ds`, `data_dg` y `filter_cb` respectivamente. Seleccione la instancia `my_ds` de DataSet en el escenario y añada las dos propiedades nuevas: `name` y `gender`. Cree una vinculación de datos entre la propiedad `dataProvider` de DataSet y la propiedad `dataProvider` del componente DataGrid mediante el panel Vinculaciones del inspector de componentes. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
my_ds.dataProvider = new Array({name:"Charles", gender:"M"}, {name:"Buddy",
  gender:"M"}, {name:"Walter", gender:"M"}, {name:"Ellen", gender:"F"},
  {name:"Jamie", gender:"F"}, {name:"Sarah", gender:"F"}, {name:"Adam",
  gender:"M"});
my_ds.addSort("nameSort", ["name"]);
my_ds.addSort("genderSort", ["gender", "name"]);
my_ds.useSort("genderSort");

filter_cb.dataProvider = [{label:"All", value:""} , {label:"Male only",
  value:"M"}, {label:"Female only", value:"F"}];
filter_cb.addEventListener("change", filterListener);
function filterListener(evt_obj:Object):Void {
  var selItem:Object = evt_obj.target.selectedItem;
  if (selItem.value.length == 0) {
    my_ds.filtered = false;
  } else {
    my_ds.filtered = true;
    my_ds.filterFunc = function(item:Object):Boolean {
      return (item.gender == selItem.value);
    }
  }
}
```

En el ejemplo siguiente se llena un componente DataGrid con contenido cargado dinámicamente mediante el componente XMLConnector. Cuando un usuario hace clic en una instancia de CheckBox en el escenario, el contenido del componente DataSet se filtra y se actualiza automáticamente en el componente DataGrid.

Arrastre los siguientes componentes al escenario y asígneles los siguientes nombres de instancia:

- CheckBox (`editorsChoice_ch`)
- DataGrid (`reviews_dg`)
- DataSet (`reviews_ds`)
- XMLConnector (`reviews_xmlconn`)

Descargue una copia del siguiente documento XML y guárdelo en su disco duro local: `www.helpexamples.com/flash/xml/reviews.xml`. Este documento XML se carga dinámicamente mediante el componente XMLConnector. Utilizará la copia local para importar el esquema XML en el componente DataSet. Seleccione la instancia de XMLConnector del escenario y seleccione la ficha Esquema del inspector de componentes. Seleccione la propiedad `results` y haga clic en Importar un esquema de un archivo XML de muestra. Seleccione el documento XML que descargó en su disco duro local y haga clic en Abrir. El esquema del documento XML debería importarse en el componente DataSet. Con la instancia de XMLConnector `reviews_xmlconn` aún seleccionada en el escenario, añada una vinculación entre la matriz `reviews_xmlconn.results.reviews.review` y la propiedad `dataProvider` de la instancia `reviews_ds` del componente DataSet. Defina la dirección de la vinculación de datos en Out en la ficha Vinculaciones. Seleccione la instancia `reviews_ds` del componente DataSet en el escenario y añada otra vinculación para la propiedad `dataProvider`. Con la instancia `reviews_ds` seleccionada, seleccione la ficha Vinculaciones del inspector de componentes. Haga clic en el botón Añadir vinculación y añada una vinculación desde la propiedad `dataProvider` del componente DataSet a la propiedad `dataProvider` de la instancia `reviews_dg` de DataGrid.

Añada el código siguiente al fotograma 1 de la línea de tiempo principal:

```
editorsChoice_ch.label = "Editor's Choice";

reviews_xmlconn.direction = "receive";
reviews_xmlconn.multipleSimultaneousAllowed = false;
reviews_xmlconn.URL = "http://www.helpexamples.com/flash/xml/reviews.xml";
reviews_xmlconn.trigger();

reviews_dg.setSize(320, 240);
reviews_dg.addColumn("name");
reviews_dg.addColumn("rating");
reviews_dg.addColumn("reviewDate");
reviews_dg.getColumnAt(0).width = 100;
```

```
reviews_dg.getColumnAt(1).width = 100;
reviews_dg.getColumnAt(2).width = 100;
reviews_dg.setStyle("alternatingRowColors", [0xFFFFFF, 0xF6F6F6]);

editorsChoice_ch.addEventListener("click", editorsChoiceListener);
function editorsChoiceListener(evt_obj:Object):Void {
  reviews_ds.filtered = evt_obj.target.selected;
  reviews_ds.filterFunc = function(item:Object):Boolean {
    return (item.editorsChoice == 1);
  };
}
```

Seleccione Control > Probar película. De forma predeterminada, el componente DataGrid debería mostrar todas la revisiones del documento XML externo. Si hace clic en el componente CheckBox elegido por el editor, se filtrará el componente DataSet y sólo se mostrarán las revisiones específicas marcadas como elegidas por el editor.

Véase también

`DataSet.filtered`

DataSet.find()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.find(`*`searchValues`*`)`

Parámetros

`searchValues` Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

Valor devuelto

Devuelve `true` si se encuentran los valores y `false` si no se encuentran.

Descripción

Método; busca en la vista actual de la colección un elemento que tenga los valores de campo especificados por *`searchValues`*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada. Si se encuentra algún elemento, éste se convierte en el elemento actual del objeto DataSet.

Los valores especificados por *searchValues* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

Si la clasificación actual no es exclusiva, el registro (objeto de transferencia) encontrado no es determinante. Si desea encontrar el primero o el último objeto de transferencia que aparezca en una clasificación no exclusiva, utilice `DataSet.findFirst()` o `DataSet.findLast()`.

La conversión de los datos especificados se realiza en función del tipo de campo subyacente. Por ejemplo, si se especifica `["05-02-02"]` como valor de búsqueda, el campo de fecha subyacente se utiliza para convertir el valor utilizando el método `DataType.setAsString()` de la fecha. Si se especifica `[new Date().getTime()]`, se utiliza el método `DataType.setAsNumber()` de la fecha.

Ejemplo

En este ejemplo se busca en la colección actual un elemento cuyos campos `name` e `id` contengan los valores `"Bobby"` y 105, respectivamente. Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo del elemento de la colección y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
student_ds.addSort("id", ["name","id"]);
// Buscar el objeto de transferencia identificado mediante "Bobby" y 105.
// Obsérvese que el orden de los campos de búsqueda coincide con
// el de los campos especificados en addSort().
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
}
// Utilizar ahora el método locateByID() para colocar el
// repetidor actual en el elemento de la colección cuyo ID coincida con
  studentID.
if (studentID != null) {
  student_ds.locateById(studentID);
}
```

Véase también

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findFirst()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.findFirst(*searchValues*)

Parámetros

searchValues Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

Valor devuelto

Devuelve `true` si se encuentran los elementos y `false` si no se encuentran.

Descripción

Método; busca en la vista actual de la colección el primer elemento que tenga los valores de campo especificados por *searchValues*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Los valores especificados por *searchValues* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

La conversión de los datos especificados se realiza en función del tipo de campo subyacente. Por ejemplo, si el valor de búsqueda especificado es `["05-02-02"]`, el campo de fecha subyacente se utiliza para convertir el valor con el método `setAsString()` de la fecha. Si el valor especificado es `[new Date().getTime()]`, se utiliza el método `setAsNumber()` de la fecha.

Ejemplo

En el siguiente ejemplo, se utiliza `DataSet.find()` para buscar en la colección actual un elemento cuyos campos `name` e `id` contengan los valores `"Bobby"` y 105, respectivamente. Si se encuentra, se utilizará `DataSet.getItemId()` para obtener un identificador exclusivo de dicho elemento y `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

Para probar este ejemplo, arrastre un componente DataSet al escenario y asígnele un nombre de instancia `student_ds`. Añada dos propiedades, `name` (tipo de datos: String) e `id` (tipo de datos: Number) al componente DataSet mediante la ficha Esquema del inspector de componentes. Si aún no tiene una copia del clip compilado `DataBindingClasses` en su biblioteca, arrastre una copia del clip compilado desde la biblioteca Clases (Ventana > Bibliotecas comunes > Clases). Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy

var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Véase también

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findLast()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.findLast(`*`searchValues`*`)`

Parámetros

`searchValues` Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

Valor devuelto

Devuelve `true` si se encuentran los elementos y `false` si no se encuentran.

Descripción

Método; busca en la vista actual de la colección el último elemento que tenga los valores de campo especificados por *`searchValues`*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Los valores especificados por *`searchValues`* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

La conversión de los datos especificados se realiza en función del tipo de campo subyacente. Por ejemplo, si el valor de búsqueda especificado es `["05-02-02"]`, el campo de fecha subyacente se utiliza para convertir el valor con el método `setAsString()` de la fecha. Si el valor especificado es `[new Date().getTime()]`, se utiliza el método `setAsNumber()` de la fecha.

Ejemplo

En el siguiente ejemplo se busca en la colección actual el último elemento cuyos campos `name` y `age` contengan `"Bobby"` y `"13"`. Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo del elemento de la colección y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
student_ds.addSort("nameAndAge", ["name", "age"]);
// Buscar el último objeto de transferencia que tenga los valores
  especificados.
// Obsérvese que el orden de los campos de búsqueda coincide con
// el de los campos especificados en addSort().
if (student_ds.findLast(["Bobby", "13"])) {
  studentID = student_ds.getItemId();
}

// Utilizar ahora el método locateByID() para colocar el
// repetidor actual en el elemento de la colección cuyo ID coincida con
  studentID.
if (studentID != null) {
  student_ds.locateById(studentID);
}
```

Véase también

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.first()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.first()

Valor devuelto

Ninguno.

Descripción

Método; convierte el primer elemento de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Ejemplo

En el siguiente código se coloca el juego de datos `inventory_ds` en el primer elemento de su colección y, a continuación, se muestra el valor de la propiedad `price` que dicho elemento contiene mediante la propiedad `DataSet.currentItem`.

```
inventory_ds.first();
trace("The price of the first item is:" + inventory_ds.currentItem.price);
```

En el siguiente ejemplo se repiten todos los elementos de la vista actual de la colección (empezando por el principio) y se realiza un cálculo en la propiedad `price` de cada elemento.

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Todo al 50%
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Véase también

`DataSet.last()`

DataSet.getItemId()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.getItemId([*index*])

Parámetros

index Número que especifica el elemento de la vista actual de elementos para obtener el ID. Este parámetro es opcional.

Valor devuelto

Una cadena.

Descripción

Método; devuelve el identificador del elemento actual de la colección o el del elemento especificado por *index*. Dicho identificador es exclusivo sólo dentro de esta colección determinada y `DataSet.addItem()` lo asigna automáticamente.

Ejemplo

El código siguiente obtiene el ID exclusivo correspondiente al elemento actual de la colección y, a continuación, lo muestra en el panel Salida.

```
var itemNo:String = my_ds.getItemId();
trace("Employee id(" + itemNo + ")");
```

En el siguiente ejemplo se utiliza `DataSet.find()` para buscar en la colección actual un elemento cuyos campos `name` e `id` contengan los valores `"Bobby"` y 105, respectivamente. Si se encuentra, se utilizará `DataSet.getItemId()` para obtener un identificador exclusivo de dicho elemento y `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

Para probar este ejemplo, arrastre un componente DataSet al escenario y asígnele un nombre de instancia `student_ds`. Añada dos propiedades, `name` (tipo de datos: String) e `id` (tipo de datos: Number) al componente DataSet mediante la ficha Esquema del inspector de componentes. Si aún no tiene una copia del clip compilado `DataBindingClasses` en su biblioteca, arrastre una copia del clip compilado desde la biblioteca Clases (Ventana > Bibliotecas comunes > Clases). Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy

var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Véase también

`DataSet.addItem()`

DataSet.getIterator()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.getIterator()`

Valor devuelto

Un objeto ValueListIterator.

Descripción

Método; devuelve un nuevo repetidor para esta colección. Dicho repetidor es una copia del repetidor actual que esté utilizándose y conserva la misma posición dentro de la colección. Este método está diseñado especialmente para usuarios avanzados que necesiten acceder a varias vistas de una misma colección de forma simultánea.

Ejemplo

En el siguiente ejemplo se utiliza `DataSet.find()` para buscar un elemento en la colección actual cuyo campo `name` contenga el valor `"Bobby"`. Aunque el repetidor `myIterator` apunta al registro de Bobby, el repetidor principal de `student_ds` sigue apuntando al último registro, Billy.

Para probar este ejemplo, arrastre un componente DataSet al escenario y asígnele un nombre de instancia `student_ds`. Añada dos propiedades, `name` (tipo de datos: String) e `id` (tipo de datos: Number) al componente DataSet mediante la ficha Esquema del inspector de componentes. Si aún no tiene una copia del clip compilado `DataBindingClasses` en su biblioteca, arrastre una copia del clip compilado desde la biblioteca Clases (Ventana > Bibliotecas comunes > Clases). Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import mx.data.to.ValueListIterator;

student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

var myIterator:ValueListIterator = student_ds.getIterator();
myIterator.sortOn(["name"]);
myIterator.find({name:"Bobby"}).id = "999";

trace(student_ds.currentItem.name + " [" + student_ds.currentItem.id +
  "]");
  // Billy [107]

student_ds.addSort("id", ["name", "id"]);
if (student_ds.find({name:"Bobby", id:999})) {
  student_ds.locateById(student_ds.getItemId());
  trace(student_ds.currentItem.name + " [" + student_ds.currentItem.id +
  "]");
    // Bobby [999]
} else {
  trace("We lost Billy!");
}
```

DataSet.getLength()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.getLength()

Valor devuelto

El número de elementos del juego de datos.

Descripción

Método; devuelve el número de elementos del juego de datos.

Ejemplo

En el siguiente ejemplo se llama a `getLength()`:

```
//...
var my_ds:mx.data.components.DataSet;
my_ds = _parent.thisShelf.compactDiscs_ds;
trace ("Data set size is: " + my_ds.getLength());
//...
```

DataSet.hasNext()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.hasNext()

Valor devuelto

Un valor booleano.

Descripción

Método; devuelve `false` si el repetidor actual se encuentra al final de la vista correspondiente de la colección y `true` en caso contrario.

Ejemplo

En el siguiente ejemplo se repiten todos los elementos de la vista actual de la colección (empezando por el principio) y se realiza un cálculo en la propiedad `price` de cada elemento.

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Todo al 50%
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Véase también

`DataSet.currentItem`, `DataSet.first()`, `DataSet.next()`

DataSet.hasPrevious()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.hasPrevious()`

Valor devuelto

Un valor booleano.

Descripción

Método; devuelve `false` si el repetidor actual se encuentra al principio de la vista correspondiente de la colección y `true` en caso contrario.

Ejemplo

En el siguiente ejemplo se repiten todos los elementos de la vista actual de la colección (empezando por el último elemento) y se realiza un cálculo en la propiedad `price` de cada elemento:

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.currentItem.price *= 0.5; // Todo al 50%
  my_ds.previous();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Véase también

`DataSet.currentItem`, `DataSet.skip()`, `DataSet.previous()`

DataSet.hasSort()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.hasSort(`*`sortName`*`)`

Parámetros

`sortName` Cadena que contiene el nombre de una clasificación creada con `DataSet.addSort()`.

Valor devuelto

Un valor booleano.

Descripción

Método; devuelve `true` si la clasificación especificada por *`sortName`* existe, y `false` en caso contrario.

Ejemplo

El código siguiente comprueba si existe una clasificación denominada “nameSort”. Si la clasificación ya existe, se convierte en la clasificación actual mediante `DataSet.useSort()`. Si no existe una clasificación con ese nombre, se creará una mediante `DataSet.addSort()`. Para probar este ejemplo, arrastre un componente List y un componente DataSet al escenario y asígneles los nombres de instancia `my_list` y `my_ds` respectivamente. Cree una vinculación de datos entre la propiedad `dataProvider` del componente DataSet y la propiedad `dataProvider` del componente List mediante el panel Vinculaciones del inspector de componentes. Cree dos propiedades en el esquema de la instancia `my_ds` de DataSet mediante la ficha Esquema del inspector de componentes: `name` (tipo de datos: String) y `years` (tipo de datos: Number). Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
import mx.data.components.datasetclasses.DataSetIterator;
my_list.labelField = "name";

my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

if (my_ds.hasSort("nameSort")) {
  my_ds.useSort("nameSort");
} else {
  my_ds.addSort("nameSort", ["name"], DataSetIterator.Descending);
}
```

Véase también

`DataSet.addSort()`, `DataSet.applyUpdates()`, `DataSet.useSort()`

DataSet.isEmpty()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.isEmpty()`

Valor devuelto

Un valor booleano.

Descripción

Método; devuelve `true` si el objeto DataSet especificado no contiene ningún elemento (es decir, si *`dataSet`*`.length == 0`).

Ejemplo

El siguiente código desactiva el botón Eliminar registro (que no se muestra) si el objeto DataSet correspondiente está vacío:

```
if (my_ds.isEmpty()) {
  delete_button.enabled = false;
}
```

Véase también

`DataSet.length`

DataSet.items

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.items`

Descripción

Propiedad; matriz de elementos administrados por *`my_ds`*.

Ejemplo

En este ejemplo se asigna una matriz de objetos a la propiedad `items` de un objeto DataSet:

```
var recData:Array = [{id:0, firstName:"Mick", lastName:"Jones"},
              {id:1, firstName:"Joe", lastName:"Strummer"},
              {id:2, firstName:"Paul", lastName:"Simonon"}];
my_ds.items = recData;
```

DataSet.itemClassName

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

*dataSet*Instance.itemClassName

Descripción

Propiedad; cadena que indica el nombre de la clase que se creará cuando se añadan elementos a la colección. La clase que se especifique debe implementar la interfaz TransferObject, tal como se muestra a continuación.

```
interface mx.data.to.TransferObject {
  function clone():Object;
  function getPropertyData():Object;
  function setPropertyData(propData:Object):Void;
}
```

También es posible establecer la propiedad en el inspector de propiedades.

Para que la clase especificada esté disponible en tiempo de ejecución, también es necesario hacer una referencia completa a la clase en algún lugar del código del archivo SWF, como se muestra en el siguiente fragmento de código:

```
var myItem:my.package.myItem;
```

Se emite la excepción `DataSetError` si se intenta modificar el valor de esta propiedad después de que se haya cargado la matriz `DataSet.items`.

Para más información, consulte "Interfaz TransferObject" en la página 1271.

DataSet.iteratorScrolled

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.iteratorScrolled = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("iteratorScrolled", listenerObject);
```

Sintaxis 2:

```
on (iteratorScrolled) {
  // ...
}
```

Descripción

Evento; se genera inmediatamente después de que el repetidor actual se haya desplazado hasta un nuevo elemento de la colección.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"iteratorScrolled"`.

`scrolled` Número que especifica la cantidad de elementos por los que se ha desplazado un repetidor; un valor positivo indica que el repetidor se ha desplazado hacia adelante en la colección, mientras que un valor negativo indica que el desplazamiento se ha realizado hacia atrás.

Ejemplo

En el siguiente ejemplo, la barra de estado de una aplicación (que no se muestra) se actualiza cuando cambia la posición del repetidor actual:

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addEventListener("iteratorScrolled", iteratorScrolledListener);

my_ds.first(); // Activar el evento iteratorScrolled.

function iteratorScrolledListener(evt_obj:Object):Void {
  trace("The iterator was scrolled.");
}
```

DataSet.last()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.last()

Valor devuelto

Ninguno.

Descripción

Método; convierte el último elemento de la vista actual de la colección en el elemento actual.

Ejemplo

El siguiente código, asociado con un componente Button, se dirige al último elemento de la colección DataSet:

```
function goLast(evt_obj:obj):Void {
  inventory_ds.last();
}
goLast_button.addEventListener("click", goLast);
```

En el siguiente ejemplo se repiten todos los elementos de la vista actual de la colección (empezando por el último elemento) y se realiza un cálculo en la propiedad `price` de cada elemento:

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.currentItem.price *= 0.5; // Todo al 50%
  my_ds.previous();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Véase también

`DataSet.first()`

DataSet.length

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.length`

Descripción

Propiedad (sólo lectura); especifica el número de elementos de la vista actual de la colección. El número de elementos que puede verse depende de la configuración actual del filtro y del rango.

Ejemplo

En el siguiente ejemplo, los eventos se desactivan antes de que los elementos de la colección sufran cambios, de forma que el rendimiento del sistema no se vea perjudicado porque el objeto DataSet intente actualizar los controles:

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.length + "
  items");
}
// Desactivar eventos para el juego de datos.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Todo al 50%
   my_ds.previous();
}

trace("After:");
traceItems();

// Indicar al juego de datos que ahora hay que actualizar los controles.
my_ds.enableEvents();

function traceItems(label:String):Void {
  for (var i:Number = 0; i < my_ds.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

DataSet.loadFromSharedObj()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.loadFromSharedObj(*objName*, [*localPath*])

Parámetros

objName Cadena que especifica el nombre del objeto compartido que debe recuperarse. El nombre puede incluir barras diagonales (/) (por ejemplo, "direcciones/trabajo"). No se permite el uso de espacios ni de los siguientes caracteres en el nombre que se especifique:

~ % & \ ; : " ' , < > ? #

localPath Parámetro de cadena opcional que especifica la ruta completa o parcial al archivo SWF que ha creado el objeto compartido. Esta cadena se utiliza para determinar la ubicación del equipo del usuario en la que se encuentra almacenado el objeto. El valor predeterminado es la ruta completa del archivo SWF.

Valor devuelto

Ninguno.

Descripción

Método; carga todos los datos necesarios para restaurar esta colección DataSet a partir de un objeto compartido. Para guardar una colección DataSet en un objeto compartido, utilice `DataSet.saveToSharedObj()`. El método `DataSet.loadFromSharedObject()` sobrescribe los datos o cambios pendientes que existan dentro de esta colección DataSet. Tenga en cuenta que el nombre de instancia de esta colección DataSet se utiliza para identificar los datos dentro de un objeto compartido determinado.

Este método emite una excepción `DataSetError` si no se encuentra el objeto compartido especificado o si surge un problema al recuperar los datos de dicho objeto.

Ejemplo

En el siguiente ejemplo se intenta cargar un objeto compartido denominado `webapp/customerInfo` asociado con el juego de datos denominado `my_ds`. La llamada al método se realiza dentro de un bloque de código `try...catch`.

```
import mx.data.components.datasetclasses.DataSetError;
try {
  my_ds.loadFromSharedObj("webapp/customerInfo");
} catch(e:DataSetError) {
  trace("Unable to load shared object.");
}
```

Véase también

`DataSet.saveToSharedObj()`

DataSet.locateById()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.locateById(*id*)

Parámetros

id Identificador de cadena del elemento de la colección que hay que buscar.

Valor devuelto

Un valor booleano.

Descripción

Método; coloca el repetidor actual en el elemento de la colección cuyo ID coincida con *id*. El método devuelve `true` si el ID especificado puede correlacionarse con un elemento de la colección, y `false` en caso contrario.

Ejemplo

En el siguiente ejemplo, se utiliza `DataSet.find()` para buscar en la colección actual un elemento cuyos campos `name` e `id` contengan los valores `"Bobby"` y 105, respectivamente. Si se encuentra, se utilizará `DataSet.getItemId()` para obtener un identificador exclusivo de dicho elemento y `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

Para probar este ejemplo, arrastre un componente DataSet al escenario y asígnele un nombre de instancia `student_ds`. Añada dos propiedades, `name` (String) e `id` (Number) al componente DataSet mediante la ficha Esquema del inspector de componentes. Si aún no tiene una copia del clip compilado `DataBindingClasses` en su biblioteca, arrastre una copia del clip compilado desde la biblioteca Clases (Ventana > Bibliotecas comunes > Clases). Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy
```

```
var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Véase también

DataSet.applyUpdates(), DataSet.find(), DataSet.getItemId()

DataSet.logChanges

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.logChanges

Descripción

Propiedad; valor booleano que especifica si deben (true) o no deben (false) registrarse en DataSet.deltaPacket los cambios realizados en el juego de datos.

Cuando esta propiedad se establece en true se registran las operaciones realizadas en los niveles de colección y elemento. Los cambios realizados en el nivel de colección incluyen las operaciones realizadas para añadir o eliminar elementos de la colección. Los cambios realizados en el nivel de elementos incluyen los cambios de propiedad realizados en los elementos y las llamadas a métodos realizadas en los elementos mediante el componente DataSet.

Ejemplo

En el ejemplo siguiente se desactiva el registro para el objeto DataSet denominado userData.

```
user_ds.logChanges = false;
```

Véase también

DataSet.deltaPacket

DataSet.modelChanged

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Descripción

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.modelChanged = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("modelChanged", listenerObject);
```

Sintaxis 2:

```
on (modelChanged) {
  // ...
}
```

Descripción

Evento; se difunde cuando se produce algún cambio en la colección; por ejemplo, cuando se añaden elementos a la colección o se eliminan elementos de ella, cuando el valor de la propiedad de un elemento varía o cuando se filtra u ordena la colección.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"iteratorScrolled"`.

`firstItem` Índice (número) del primer elemento de la colección afectado por el cambio.

`lastItem` Índice (número) del último elemento de la colección afectado por el cambio (es igual a `firstItem` si sólo se ve afectado un elemento).

`fieldName` Cadena que contiene el nombre del campo afectado. Esta propiedad no está definida (es decir, tiene un valor de `undefined`) salvo si el cambio se ha efectuado en una propiedad del objeto DataSet.

`eventName` Cadena que describe el cambio realizado. Puede tratarse de cualquiera de los valores siguientes:

Valor de la cadena	Descripción
`"addItems"`	Se ha añadido una serie de elementos.
`"filterModel"`	El modelo se ha filtrado y es necesario actualizar la vista (restablecer la posición de desplazamiento)
`"removeItems"`	Se ha eliminado una serie de elementos.
`"schemaLoaded"`	Se ha declarado la definición de los campos del proveedor de datos.
`"sort"`	Se han ordenado los datos.
`"updateAll"`	Es necesario actualizar toda la vista, excluida la posición de desplazamiento.
`"updateColumn"`	Es necesario actualizar la definición de la totalidad de un campo del proveedor de datos.
`"updateField"`	Un campo de un elemento ha cambiado y debe actualizarse.
`"updateItems"`	Es necesario actualizar una serie de elementos.

Ejemplo

En el siguiente ejemplo, se distribuye el evento modelChanged siempre que se añade o se elimina un elemento del juego de datos:

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("[event =" + evt_obj.eventName + "] the data set now has " +
  evt_obj.target.items.length + " items.");
}
my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});
my_ds.removeItemAt(0);
```

En el siguiente ejemplo, el botón que sirve para eliminar elementos está desactivado si se han eliminado los elementos de la colección y si el objeto DataSet de destino no tiene más elementos:

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Desactivar eventos para el juego de datos.
my_ds.disableEvents();
```

```
my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while (my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Todo al 50%
   my_ds.previous();
}

trace("After:");
traceItems();

// Indicar al juego de datos que ahora hay que actualizar los controles.
my_ds.enableEvents();

function traceItems(label:String):Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Véase también

DataSet.isEmpty()

DataSet.newItem

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.newItem = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("newItem", listenerObject);
```

Sintaxis 2:

```
on (newItem) {
  // ...
}
```

Descripción

Evento; se difunde cuando se construye un nuevo objeto de transferencia mediante `DataSet.createItem()`. Un detector de este evento puede realizar modificaciones en el elemento antes de añadirlo a la colección.

El objeto de evento (*`eventObj`*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"iteratorScrolled"`.

`item` Referencia al elemento creado.

Ejemplo

En el siguiente ejemplo se realizan modificaciones en un elemento recién creado antes de añadirlo a la colección:

```
function newItemEvent(evt_obj:Object):Void {
  var employee:Object = evt_obj.item;
  employee.name = "newGuy";
  // Los datos de la propiedad son XML.
  employee.zip =
  employee.getPropertyData().firstChild.childNodes[1].attributes.zip;
}
employees_ds.addEventListener("newItem", newItemEvent);
```

DataSet.next()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.next()`

Valor devuelto

Ninguno.

Descripción

Método; convierte el siguiente elemento de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Ejemplo

En el siguiente ejemplo se repiten todos los elementos de la vista actual de la colección (empezando por el principio) y se realiza un cálculo en la propiedad `price` de cada elemento:

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Todo al 50%
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Véase también

`DataSet.first()`, `DataSet.hasNext()`

DataSet.previous()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.previous()`

Valor devuelto

Ninguno.

Descripción

Método; convierte el elemento anterior de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Ejemplo

En el siguiente ejemplo se repite indefinidamente cada elemento de un juego de datos y se rastrea el precio de cada elemento:

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});
my_ds.last();
while (my_ds.hasPrevious()) {
  trace(my_ds.currentItem.price);
  my_ds.previous();
}
```

En el siguiente ejemplo se repiten indefinidamente todos los elementos de la vista actual de la colección, empezando por el último, y se realiza un cálculo en un campo de cada elemento:

```
my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.price *= 0.5; // Todo al 50%
  my_ds.previous();
}
```

Véase también

`DataSet.first()`, `DataSet.hasNext()`

DataSet.properties

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.properties`

Descripción

Propiedad (sólo lectura); devuelve un objeto que contiene todas las propiedades expuestas (campos) correspondientes a cualquier objeto de transferencia que se encuentre dentro de esta colección.

Ejemplo

En el siguiente ejemplo se muestran todos los nombres de las propiedades del objeto DataSet denominado `my_ds`:

```
var i:String;
for (i in my_ds.properties) {
  trace("field '" + i + "' has value " + my_ds.properties[i]);
}
```

DataSet.readOnly

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.readOnly`

Descripción

Propiedad; valor booleano que especifica si la colección puede modificarse (`false`) o es de sólo lectura (`true`). Si la propiedad se establece en `true` no podrán realizarse actualizaciones de la colección. El valor predeterminado es `false`.

También es posible establecer la propiedad en el inspector de propiedades.

Ejemplo

En el ejemplo siguiente se convierte el objeto DataSet denominado `my_ds` en un objeto de sólo lectura y, a continuación, se intenta modificar el valor de una propiedad que pertenece al elemento actual de la colección. Este intento emite una excepción `DataSetError`.

```
import mx.data.components.datasetclasses.DataSetError;
my_ds.readOnly = true;
try {
  // Con esta acción se emitirá una excepción.
  my_ds.addItem({name:'Joe'});
} catch (e:DataSetError) {
  // Clasificación especificada 'name' no existe para DataSet 'my_ds'.
  trace("DataSetError >> " + e.message);
}
```

Véase también

`DataSet.currentItem`

DataSet.removeAll()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
dataSetInstance.removeAll()
```

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la colección DataSet.

Ejemplo

En el siguiente ejemplo se eliminan todos los elementos de la colección DataSet `contact_ds`:

```
contact_ds.removeAll();
```

DataSet.removeItem

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.removeItem = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("removeItem", listenerObject);
```

Sintaxis 2:

```
on (removeItem) {
  // ...
}
```

Descripción

Evento; generado justo antes de eliminar un nuevo elemento de esta colección.

Si se especifica el valor `false` en la propiedad `result` del objeto en el que se ha producido el evento, se cancelará la operación de eliminación; en cambio, si se establece el valor `true`, podrá realizarse dicha operación.

El objeto de evento (*`eventObj`*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"removeItem"`.

`item` Referencia al elemento de la colección que se va a eliminar.

`result` Valor booleano que especifica si el elemento debe o no eliminarse. De forma predeterminada, este valor es `true`.

Ejemplo

En el siguiente ejemplo, un controlador de eventos `on(removeItem)` cancela la eliminación del nuevo elemento si una función definida por el usuario denominada `userHasAdminPrivs()` devuelve `false`; de lo contrario se permite llevar a cabo la eliminación:

```
on (removeItem) {
  if (globalObj.userHasAdminPrivs()) {
    // Permitir la eliminación del elemento.
    eventObj.result = true;
  } else {
    // No permitir la eliminación del elemento; el usuario no tiene
  privilegios de administrador.
    eventObj.result = false;
  }
}
```

El controlador de eventos `addItem` siguiente cancela la eliminación del elemento existente si una función definida por el usuario denominada `userHasAdminPrivs()` devuelve el valor `false`; de lo contrario, sí se admite la eliminación.

```
function userHasAdminPrivs():Boolean {
  return false; // cambiar esto a true para permitir inserciones
}
function removeItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Allow the item removal.
    evt_obj.result = true;
    trace("Item removed");
  } else {
    // No permitir la eliminación del elemento; el usuario no tiene
  privilegios de administrador.
    evt_obj.result = false;
    trace("Error, insufficient permissions");
  }
}
my_ds.addEventListener("removeItem", removeItemListener);
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});
my_ds.removeItemAt(0);
```

Véase también

`DataSet.addItem`

DataSet.removeItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.removeItem([`*`item`*`])`

Parámetros

`item` Elemento que debe eliminarse. Este parámetro es opcional.

Valor devuelto

Un valor booleano. Devuelve `true` si el elemento se eliminó con éxito, y `false` en caso contrario.

Descripción

Método; elimina el elemento especificado de la colección, o bien elimina el elemento actual si se omite el parámetro *item*. Está operación se registra en DataSet.deltaPacket si el valor de DataSet.logChanges es true.

Ejemplo

El código siguiente elimina el elemento de la posición actual del repetidor. Para probar este ejemplo, añada un componente DataSet al escenario y asígnele un nombre de instancia my_ds. Añada el código siguiente al fotograma 1 de la línea de tiempo principal:

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

trace(my_ds.getLength()); // 5
trace(my_ds.currentItem.name); // Frank
my_ds.removeItem();
trace(my_ds.getLength()); // 4
trace(my_ds.currentItem.name); // Michael
```

Véase también

DataSet.deltaPacket, DataSet.logChanges

DataSet.removeItemAt()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

dataSetInstance.removeItemAt(*index*)

Parámetros

index Número mayor o igual que 0. Este número es el índice del elemento que se va a eliminar.

Valor devuelto

Un valor booleano que indica si el elemento se ha eliminado o no.

Descripción

Método; elimina el elemento del índice especificado. Los índices que se encuentran después del índice eliminado se contraerán una posición.

Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

Además, activa el evento DataSet.removeItem, que contiene las propiedades `result` y `item`. La propiedad `result` se utiliza para determinar si el elemento al que hace referencia la propiedad `item`del evento puede eliminarse. De forma predeterminada, la propiedad `result` está establecida en `true`. Si no se especifica ningún detector de eventos para el evento `removeItem`, el elemento se elimina de forma predeterminada.

Un detector de eventos puede evitar que el elemento se elimine mediante la detección del evento `removeItem` y la configuración de la propiedad `result` del evento en `false`, como se muestra en el siguiente ejemplo:

```
function removeItem(evt_obj:Object):Void {
  // No permitir a nadie eliminar el elemento con customerId == 0.
  evt_obj.result = (evt_obj.item.customerId != 0);
}
```

Ejemplo

En el siguiente ejemplo se elimina un elemento del juego de datos en la primera posición:

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

trace(my_ds.getLength()); // 5
trace(my_ds.currentItem.name); // Frank
my_ds.removeItemAt(0);
trace(my_ds.getLength()); // 4
trace(my_ds.currentItem.name); // Frank
```

DataSet.removeRange()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.removeRange()`

Valor devuelto

Ninguno.

Descripción

Método; elimina la configuración especificada del punto final actual mediante `DataSet.setRange()` para el repetidor actual.

Ejemplo

```
my_ds.addSort("name_id", ["name", "id"]);
my_ds.setRange(["Bobby", 105],["Cathy", 110]);
while (my_ds.hasNext()) {
  my_ds.gradeLevel ="5"; // Cambiar todas las calificaciones de este rango.
  my_ds.next();
}
my_ds.removeRange();
my_ds.removeSort("name_id");
```

Véase también

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeSort()`, `DataSet.setRange()`

DataSet.removeSort()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.removeSort(`*`sortName`*`)`

Parámetros

`sortName` Cadena que especifica el nombre de la clasificación que va a eliminarse.

Valor devuelto

Ninguno.

Descripción

Método; elimina la clasificación especificada de este objeto DataSet si dicha clasificación existe. Si la clasificación especificada no existe, el método emitirá una excepción `DataSetError`.

Ejemplo

En el siguiente ejemplo se crea un rango de elementos en el componente DataSet y se modifica la propiedad gradeLevel de cada elemento. Para probar este ejemplo, añada un componente DataSet al escenario y asígnele un nombre de instancia `my_ds`. Con el componente DataSet seleccionado, cree tres propiedades nuevas en el esquema del componente DataSet mediante la ficha Esquema del inspector de componentes. Asigne a las nuevas propiedades los nombres `name`, `id` y `gradeLevel` y los tipos de datos String, Number y Number, respectivamente. Añada una copia del clip compilado `DataBindingClasses` desde la biblioteca común Clases (Ventana > Bibliotecas comunes > Clases) y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});

my_ds.addSort("name_id", ["name", "id"]);
my_ds.setRange(["Bobby", 105], ["Cathy", 110]);
while (my_ds.hasNext()) {
  my_ds.gradeLevel = "5"; // Cambiar todas las calificaciones de este
  rango.
  my_ds.next();
}
my_ds.removeRange();
my_ds.removeSort("name_id");

for (var i=0; i<my_ds.length; i++) {
  trace(my_ds.items[i].name + " > " + my_ds.items[i].gradeLevel);
}
```

Véase también

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeRange()`, `DataSet.setRange()`

DataSet.resolveDelta

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.resolveDelta = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("resolveDelta", listenerObject);
```

Sintaxis 2:

```
on (resolveDelta) {
  // ...
}
```

Descripción

Evento; se difunde cuando se asigna un paquete delta a `DataSet.deltaPacket`, cuyo ID de transacción coincide con el de un paquete delta previamente recuperado a partir del objeto DataSet, y que tiene mensajes asociados con cualquiera de los objetos Delta o DeltaItem que contiene el paquete delta.

Este evento permite enmendar los errores devueltos desde el servidor al intentar aplicar los cambios enviados anteriormente. Normalmente, este evento se utiliza para mostrar un cuadro de diálogo de soluciones con los valores incorrectos, de forma que el usuario pueda realizar las modificaciones pertinentes en los datos y los pueda volver a enviar.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

`target` Objeto DataSet que generó el evento.

`type` Cadena `"resolveDelta"`.

`data` Matriz de deltas y objetos DeltaItem asociados con mensajes de longitud distinta de cero.

Ejemplo

En el siguiente ejemplo se muestra un formulario denominado `reconcileForm` (que no se muestra) y se llama a un método de ese objeto Form (`setReconcileData()`) que permite al usuario corregir los valores devueltos como incorrectos por el servidor:

```
import mx.data.components.datasetclasses.*;
my_ds.addEventListener("resolveDelta", onResolveDelta);
function onResolveDelta(eventObj:Object) {
  reconcileForm.visible = true;
  reconcileForm.setReconcileData(eventObj.data);
}
// en el código de reconcileForm
function setReconcileData(data:Array):Void {
  var di:DeltaItem;
  var ops:Array = ["property", "method"];
  var cl:Array;
  // change list
  var msg:String;
  for (var i = 0; i<data.length; i++) {
    cl = data[i].getChangeList();
    for (var j = 0; j<cl.length; j++) {
      di = cl[j];
      msg = di.message;
      if (msg.length>0) {
        trace("The following problem occurred '"+msg+"' while performing a
  '"+ops[di.kind]+"' modification on/with '"+di.name+"' current server
  value ["+di.curValue+"], value sent ["+di.newValue+"] Please fix!");
      }
    }
  }
}
```

DataSet.saveToSharedObj()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.saveToSharedObj(*objName*, [*localPath*])

Parámetros

objName Cadena que especifica el nombre del objeto compartido que va a crearse. El nombre puede incluir barras diagonales (/) (por ejemplo, “direcciones/trabajo”). No se permite el uso de espacios ni de los siguientes caracteres en el nombre que se especifique:

~ % & \ ; : " ' , < > ? #

localPath Parámetro de cadena opcional que especifica la ruta completa o parcial al archivo SWF que ha creado el objeto compartido. Esta cadena se utiliza para determinar la ubicación del equipo del usuario en la que se encuentra almacenado el objeto. El valor predeterminado es la ruta completa del archivo SWF.

Valor devuelto

Ninguno.

Descripción

Método; guarda todos los datos necesarios para restaurar esta colección DataSet en un objeto compartido. De esta forma los usuarios pueden trabajar aunque estén desconectados de los datos de origen, si se trata de recursos de red. Este método sobrescribe los datos que existen dentro del objeto de datos compartido de esta colección DataSet. Para restaurar una colección DataSet desde un objeto compartido, utilice `DataSet.loadFromSharedObj()`. Tenga en cuenta que el nombre de esta colección DataSet se utiliza para identificar los datos dentro de un objeto compartido determinado.

Si no puede crearse el objeto compartido o si se produce un problema al alinear los datos a éste, el método emitirá la excepción `DataSetError`.

Ejemplo

En el siguiente ejemplo se llama a `saveToSharedObj()` en un bloque `try..catch` y se muestra un error si se produce algún problema al guardar los datos en el objeto compartido.

```
import mx.data.components.datasetclasses.DataSetError;
try {
  my_ds.saveToSharedObj("webapp/customerInfo");
} catch(e:DataSetError) {
  trace("Unable to create shared object");
}
```

Véase también

`DataSet.loadFromSharedObj()`

DataSet.schema

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.schema

Descripción

Propiedad; proporciona la representación en XML del esquema para este objeto DataSet. El código XML asignado a esta propiedad debe tener el formato siguiente:

```
<?xml version="1.0"?>
<properties>
  <property name="propertyName">
    <type name="dataType" />
    <encoder name="dataType">
      <options>
        <dataFormat>format options<dataFormat/>
      <options/>
    <encoder/>
    <kind name="dataKind">
    </kind>
  </property>
  <property> ... </property>
  ...
</properties>
```

Se emitirá una excepción `DataSetError` si el código XML especificado no respeta el formato anterior.

Ejemplo

En el siguiente ejemplo se establece el esquema del juego de datos `my_ds` a un nuevo objeto XML que contiene XML con formato apropiado:

```
my_ds.schema = new XML("<properties><property name="billable"> ..etc.. </
  properties>");
```

DataSet.selectedIndex

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.selectedIndex

Descripción

Propiedad; especifica el índice seleccionado dentro de la colección. Esta propiedad puede vincularse al elemento seleccionado en un componente DataGrid o List, y viceversa. Si desea ver un ejemplo completo de demostración, consulte "Creación de aplicaciones con el componente DataSet" en la página 349.

Ejemplo

En el siguiente ejemplo se establece el índice seleccionado de un objeto DataSet (`user_ds`) en el índice seleccionado de un componente DataGrid (`user_dg`).

```
user_ds.selectedIndex = user_dg.selectedIndex;
```

DataSet.setIterator()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.setIterator(*iterator*)

Parámetros

iterator Objeto repetidor devuelto por una llamada a `DataSet.getIterator()`.

Valor devuelto

Ninguno.

Descripción

Método; asigna el repetidor especificado a este objeto DataSet y lo convierte en el repetidor actual. El repetidor especificado debe provenir de una llamada anterior a `DataSet.getIterator()` en el objeto DataSet al que se asigne; de lo contrario, se emitirá una excepción `DataSetError`.

Ejemplo

```
import mx.data.to.ValueListIterator;
myIterator:ValueListIterator = my_ds.getIterator();
myIterator.sortOn(["name"]);
my_ds.setIterator(myIterator);
```

Véase también

`DataSet.getIterator()`

DataSet.setRange()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.setRange(`*`startValues`*`, `*`endValues`*`)`

Parámetros

`startValues` Matriz de valores clave de las propiedades del primer objeto de transferencia del rango.

`endValues` Matriz de valores clave de las propiedades del último objeto de transferencia del rango.

Valor devuelto

Ninguno.

Descripción

Método; establece los puntos finales del repetidor actual. Los puntos finales definen un rango dentro del cual opera el repetidor. Sólo es válido si se ha establecido una clasificación válida para el repetidor actual mediante `DataSet.addSort()`.

Establecer un rango para el repetidor actual es más eficaz que utilizar una función de filtro si lo que se desea es agrupar los valores (consulte `DataSet.filterFunc`).

Ejemplo

En el siguiente ejemplos e selecciona un rango de estudiantes y se localiza cada uno de sus nombres en el panel Salida:

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});
my_ds.addSort("name_id",["name", "id"]);
my_ds.setRange(["Bobby", 105],["Cathy", 110]);
while (my_ds.hasNext()) {
  trace(my_ds.name); // Bobby..Cathy
  my_ds.next();
}
```

Véase también

`DataSet.addSort()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeRange()`, `DataSet.removeSort()`

DataSet.skip()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`dataSetInstance``.skip(`*`offSet`*`)`

Parámetros

`offSet` Número entero que especifica el número de registros que se desplazará la posición del repetidor.

Valor devuelto

Ninguno.

Descripción

Método; desplaza la posición del repetidor actual hacia delante o hacia atrás en función del número especificado por *offSet*. Los valores positivos de *offSet* desplazan la posición del repetidor actual hacia delante, mientras que los negativos la desplazan hacia atrás.

Si el valor de offset especificado se encuentra fuera del principio (o del final) de la colección, el repetidor se colocará al principio (o al final) de la colección.

Ejemplo

En el siguiente ejemplo, se coloca el repetidor actual en el primer elemento de la colección, a continuación se desplaza al penúltimo elemento y por último se realiza un cálculo en un campo de dicho elemento:

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.first();
var itemsToSkip:Number = 3;
trace(my_ds.currentItem.name); // Billy
my_ds.skip(itemsToSkip);
trace(my_ds.currentItem.name); // Mally
```

DataSet.useSort()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

dataSetInstance.useSort(*sortName*, *order*)

Parámetros

sortName Cadena que contiene el nombre de la clasificación que va a utilizarse.

order Valor entero que indica el orden de la clasificación; el valor debe ser `DataSetIterator.Ascending` o `DataSetIterator.Descending`.

Valor devuelto

Ninguno.

Descripción

Método; cambia la clasificación del repetidor actual a la que se especifica en *sortName*, si existe. Si la clasificación especificada no existe, se emite una excepción `DataSetError`.

Para crear una clasificación, utilice `DataSet.addSort()`.

Ejemplo

En el siguiente ejemplo se utiliza `DataSet.hasSort()` para determinar si una clasificación denominada `"customer"` existe o no. Si existe, el código llama a `DataSet.useSort()` para que `"customer"` sea la clasificación actual. De lo contrario, el código crea una clasificación con ese nombre mediante `DataSet.addSort()`.

```
if (my_ds.hasSort("customer")) {
  my_ds.useSort("customer");
} else {
  my_ds.addSort("customer", ["customer"], DataSetIterator.Descending);
}
```

Véase también

`DataSet.applyUpdates()`, `DataSet.hasSort()`

CAPÍTULO 14

Componente DateChooser (sólo en Flash Professional)

14

El componente DateChooser es un calendario que permite a los usuarios seleccionar una fecha. Tiene botones que permiten al usuario desplazarse por los meses y hacer clic en cualquier fecha para seleccionarla. Pueden establecerse parámetros que indican el nombre del mes y del día, el primer día de la semana o las fechas desactivadas. También es posible resaltar la fecha actual.

La previsualización dinámica de cada instancia de DateChooser refleja los valores indicados durante la edición en el inspector de propiedades o el inspector de componentes.

Utilización del componente DateChooser (sólo en Flash Professional)

DateChooser puede utilizarse siempre que se desee que un usuario seleccione una fecha. Por ejemplo, podría utilizarse el componente DateChooser en un sistema de reservas de hotel en el que unas fechas determinadas puedan seleccionarse y otras queden desactivadas. También podría utilizarse el componente DateChooser en una aplicación que muestre, cuando el usuario elija una fecha, los eventos en curso, como por ejemplo actuaciones o reuniones.

Parámetros de DateChooser

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DateChooser en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

dayNames establece el nombre de los días de la semana. El valor es una matriz y el valor predeterminado es `["S", "M", "T", "W", "T", "F", "S"]`.

disabledDays indica los días de la semana que están desactivados. Este parámetro es una matriz que puede tener hasta siete valores. El valor predeterminado es `[]` (matriz vacía).

firstDayOfWeek indica el día de la semana que se muestra en la primera columna de selector de fechas (0-6, con 0 como primer elemento de la matriz de `dayNames`). Esta propiedad cambia el orden de visualización de las columnas de los días.

monthNames establece el nombre de los meses que se muestran en la fila de encabezado del calendario. El valor es una matriz y el valor predeterminado es `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"]`.

showToday indica si debe resaltarse la fecha de hoy o no. El valor predeterminado es `true`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente DateChooser en el inspector componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no producirá ningún efecto visible.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de DateChooser utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase DateChooser (sólo en Flash Professional)" en la página 432.

Creación de aplicaciones con el componente DateChooser

El procedimiento siguiente explica cómo añadir un componente DateChooser a una aplicación durante la edición. En este ejemplo, el selector de fecha permite al usuario elegir una fecha en un sistema de reservas de una compañía aérea. Todas las fechas anteriores al 15 de octubre deben quedar desactivadas. También debe desactivarse un rango de días de diciembre, de manera que se cree un período de vacaciones que quede excluido; además los lunes deben quedar desactivados.

Para crear una aplicación con el componente DateChooser:

1. En el panel Componentes haga doble clic en el componente DateChooser para añadirlo al escenario.

2. En el inspector de propiedades, introduzca el nombre de instancia **flightCalendar**.
3. En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas seleccionables:

```
flightCalendar.selectableRange = {rangeStart:new Date(2003, 9, 15),
  rangeEnd:new Date(2003, 11, 31)}
```

   Este código asigna un valor a la propiedad `selectableRange` de un objeto ActionScript que contenga dos objetos Date con los nombres de variable `rangeStart` y `rangeEnd`. De esta forma se definen los límites superior e inferior de un rango dentro del cual el usuario puede seleccionar una fecha.
4. En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas desactivadas de días festivos:

```
flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
  rangeEnd: new Date(2003, 11, 26)}];
```

5. En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para que los lunes queden desactivados:

```
flightCalendar.disabledDays=[1];
```

6. Seleccione Control > Probar película.

Para crear una instancia del componente DateChooser mediante código ActionScript:

1. Arrastre el componente DateChooser desde el panel Componentes a la biblioteca del documento actual.
2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
this.createClassObject(mx.controls.DateChooser, "my_dc", 1);
```

   Este script utiliza el método "UIObject.createClassObject()" en la página 1404 para crear la instancia DateChooser y, a continuación, cambia de tamaño y coloca la cuadrícula.
3. Seleccione Control > Probar película.

Personalización del componente DateChooser (sólo en Flash Professional)

El componente DateChooser puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`).

Utilización de estilos con el componente DateChooser

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de DateChooser. Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente DateChooser admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Color de la iluminación cuando las fechas están seleccionadas y se pasa el puntero sobre ellas. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Ambos	Color del fondo. El valor predeterminado es 0xEFEBEF (gris claro).
`borderColor`	Ambos	Color del borde. El valor predeterminado es 0x919999. El componente DateChooser utiliza una línea sólida de un solo píxel como borde. Este borde no puede modificarse con estilos o aspectos.
`headerColor`	Ambos	El color de fondo para el encabezado del componente. El color predeterminado es blanco.
`rollOverColor`	Ambos	Color de fondo de una fecha por la que se desplaza el puntero. El valor predeterminado es 0xE3FFD6 (verde brillante) con el tema Halo y 0xAAAAAA (gris claro) con el tema Sample.

Estilo	Tema	Descripción
`selectionColor`	Ambos	Color de fondo de la fecha seleccionada. El valor predeterminado es 0xCDFFC1 (verde claro) con el tema Halo y 0xEEEEEE (gris muy claro) con el tema Sample.
`todayColor`	Ambos	El color de fondo para la fecha de hoy. El valor predeterminado es 0x666666 (gris oscuro).
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

El componente DateChooser utiliza cuatro categorías de texto para mostrar el nombre del mes, los días de la semana, la fecha de hoy y las fechas habituales. Las propiedades de estilo de texto establecidas en el componente DateChooser controlan el texto de la fecha habitual y proporcionan estilos predeterminados para los demás textos. Si se desea establecer estilos de texto para categorías de texto específicas, es necesario utilizar las siguientes declaraciones de estilo de clase.

Nombre de la declaración	Descripción
`HeaderDateText`	Nombre del mes.
`WeekDayStyle`	Días de la semana.
`TodayStyle`	Fecha de hoy.

En el siguiente ejemplo se muestra cómo establecer el nombre del mes y los días de la semana en rojo oscuro.

```
_global.styles.HeaderDateText.setStyle("color", 0x660000);
_global.styles.WeekDayStyle.setStyle("color", 0x660000);
```

Utilización de aspectos con el componente DateChooser

El componente DateChooser utiliza aspectos para representar los botones de mes siguiente y anterior y el indicador de fecha de hoy. Para aplicar un aspecto al componente DateChooser durante la edición, modifique los símbolos de aspecto en la carpeta Flash UI Components 2/Themes/MMDefault/DateChooser Assets/States de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

En este componente sólo se pueden aplicar aspectos a los botones de desplazamiento de mes. Un componente DateChooser utiliza las propiedades de aspecto siguientes:

Propiedad	Descripción
`backMonthButtonUpSymbolName`	Estado sin presionar del botón de mes anterior. El valor predeterminado es `backMonthUp`.
`backMonthButtonDownSymbolName`	Estado presionado del botón de mes anterior. El valor predeterminado es `backMonthDown`.
`backMonthButtonDisabledSymbolName`	Estado desactivado del botón de mes anterior. El valor predeterminado es `backMonthDisabled`.

Propiedad	Descripción
`fwdMonthButtonUpSymbolName`	Estado sin presionar del botón de mes siguiente. El valor predeterminado es `fwdMonthUp`.
`fwdMonthButtonDownSymbolName`	Estado presionado del botón de mes siguiente. El valor predeterminado es `fwdMonthDown`.
`fwdMonthButtonDisabledSymbolName`	Estado desactivado del botón de mes siguiente. El valor predeterminado es `fwdMonthDisabled`.

Los símbolos de botón se utilizan exactamente así, sin aplicar colores ni cambiar tamaños. El tamaño lo determina el símbolo durante la edición.

Para crear símbolos de clip de película para aspectos de DateChooser:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta DateChooser Assets a la biblioteca del documento.
4. Expanda la carpeta DateChooser Assets/States de la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo backMonthDown.
6. Personalice el símbolo como desee.

   Por ejemplo, cambie el color de la flecha a rojo.
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.

   Por ejemplo, cambie el color del símbolo abajo de flecha siguiente para que coincida con la flecha anterior.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente DateChooser al escenario.

10. Seleccione Control > Probar película.

NOTA

La carpeta DateChooser Assets/States también contiene una carpeta de aspectos diarios con un único elemento de aspecto, `cal_todayIndicator`. Este elemento puede modificarse durante la edición para personalizar el indicador de fecha de hoy. Sin embargo, no puede cambiarse dinámicamente en una instancia de DataChooser particular para utilizar un símbolo diferente. Además, el símbolo `cal_todayIndicator` debe ser un gráfico de un único color sólido porque el componente DateChooser aplicará el color `todayColor` al gráfico como un todo. El gráfico puede tener recortes, pero hay que recordar que el color de texto predeterminado para la fecha de hoy es el blanco y que el color de fondo para el componente DateChooser es el blanco, por lo que un recorte en mitad del elemento de aspecto del indicador de fecha de hoy haría imposible la lectura de la fecha de hoy, a menos que se cambiara el color de fondo o el color de texto de la misma.

Clase DateChooser (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > DateChooser

Nombre de clase de ActionScript mx.controls.DateChooser

Las propiedades de la clase DateChooser permiten acceder a la fecha seleccionada y al mes y día mostrados. También se puede indicar el nombre de los días y los meses, las fechas desactivadas y seleccionables, establecer el primer día de la semana y decidir si debe resaltarse o no la fecha actual.

Si una propiedad de la clase DateChooser se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.DateChooser.version);
```

NOTA

El código `trace(myDC.version);` devuelve `undefined`.

Resumen de métodos de la clase DateChooser

No hay métodos exclusivos de la clase DateChooser.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase DateChooser de la clase UIObject. Al llamar a estos métodos desde el objeto DateChooser, debe utilizarse la forma *dateChooserInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase DateChooser de la clase UIComponent. Al llamar a estos métodos desde el objeto DateChooser, debe utilizarse la forma *dateChooserInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase DateChooser

En la siguiente tabla se enumeran las propiedades que son exclusivas de la clase DataChooser.

Propiedad	Descripción
`DateChooser.dayNames`	Matriz que indica el nombre de los días de la semana.
`DateChooser.disabledDays`	Matriz que indica los días de la semana desactivados para todas las fechas aplicables del selector de fechas.
`DateChooser.disabledRanges`	Rango de fechas desactivadas o fecha única desactivada.
`DateChooser.displayedMonth`	Número que indica un elemento de la matriz `monthNames` para mostrarlo en el selector de fechas.
`DateChooser.displayedYear`	Número que indica el año que se mostrará.
`DateChooser.firstDayOfWeek`	Número que indica un elemento de la matriz `dayNames` que se mostrará en la primera columna del selector de fechas.
`DateChooser.monthNames`	Matriz de cadenas que indica el nombre de los meses.
`DateChooser.selectableRange`	Fecha única seleccionable o rango de fechas seleccionables.
`DateChooser.selectedDate`	Objeto Date que indica la fecha seleccionada.
`DateChooser.showToday`	Valor booleano que indica si debe resaltarse o no la fecha actual.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase DateChooser de la clase UIObject. Al acceder a estas propiedades desde el objeto DateChooser, debe utilizarse la forma *`dateChooserInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase DateChooser de la clase UIComponent. Al acceder a estas propiedades desde el objeto DateChooser, debe utilizarse la forma *`dateChooserInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase DateChooser

En la siguiente tabla se enumeran los eventos que son exclusivos de la clase DataChooser.

Evento	Descripción
`DateChooser.change`	Se difunde cuando se selecciona una fecha.
`DateChooser.scroll`	Se difunde cuando se hace clic en los botones de mes.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase DateChooser de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase DateChooser de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

DateChooser.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
dateChooserInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  //...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una fecha.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*dataChooserInstance*) distribuye un evento (en este caso, `change`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de DateChooser. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `my_dc`, envía "_level0.my_dc" al panel Salida:

```
on (change) {
  trace(this);
}
```

Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se cambia una instancia de DateChooser denominada `my_dc`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `change` en el objeto detector. Dentro de la función hay una sentencia `trace()` que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto de evento es el componente que ha generado el evento (en este ejemplo `my_dc`).

```
// Crear un objeto detector.
var dcListener:Object = new Object();
dcListener.change = function(evt_obj:Object) {
 var thisDate:Date = evt_obj.target.selectedDate;
    trace("date selected: " +  thisDate);
};

// Añadir objeto detector al selector de fecha.
my_dc.addEventListener("change", dcListener);
```

DateChooser.dayNames

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateChooserInstance``.dayNames`

Descripción

Propiedad; matriz que contiene el nombre de los días de la semana. El domingo es el primer día (en la posición de índice 0) y el resto de los nombres de día siguen en el orden habitual. El valor predeterminado es `["S", "M", "T", "W", "T", "F", "S"]`.

Ejemplo

En el siguiente ejemplo cambia el valor de los días de la semana:

```
my_dc.dayNames = new Array("Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");
```

DateChooser.disabledDays

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateChooserInstance``.disabledDays`

Descripción

Propiedad; matriz que indica cuáles son los días de la semana que están desactivados. Todas las fechas del mes que coincidan con el día especificado quedarán desactivadas. Los valores de los elementos de esta matriz pueden estar comprendidos entre el 0, para el domingo, y el 6, para el sábado. El valor predeterminado es `[]` (matriz vacía).

Ejemplo

En el ejemplo siguiente se desactivan los domingos y los sábados de forma que los usuarios sólo puedan seleccionar el resto de los días:

```
my_dc.disabledDays = [0, 6];
```

DateChooser.disabledRanges

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.disabledRanges

Descripción

Propiedad; desactiva un único día o un rango de días. Esta propiedad es una matriz de objetos. Cada objeto de la matriz debe ser un objeto Date, que especifique una fecha única para desactivarla, o bien un objeto que contenga las propiedades `rangeStart` y `rangeEnd` (o al menos una de ellas), cuyos valores deben ser un objeto Date. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas. Si se omite alguna de dichas propiedades, el rango quedará abierto por la fecha correspondiente.

El valor predeterminado de `disabledRanges` es `undefined`.

Especifique una fecha completa al definir fechas para la propiedad `disabledRanges`. Por ejemplo, especifique `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, el objeto Date devuelve la fecha y hora actuales. Si no se especifica una hora, la hora devuelta es `00:00:00`.

Ejemplo

En el ejemplo siguiente se define una matriz con los objetos Date `rangeStart` y `rangeEnd`, que desactivan las fechas comprendidas entre el 7 de mayo y el 7 de junio:

```
my_dc.disabledRanges = [{rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

En el ejemplo siguiente se desactivan todas las fechas posteriores al 7 de noviembre:

```
my_dc.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

En el ejemplo siguiente se desactivan todas las fechas anteriores al 7 de octubre:

```
my_dc.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

En el ejemplo siguiente se desactiva únicamente el 7 de diciembre:

```
my_dc.disabledRanges = [ new Date(2003, 11, 7) ];
```

En el siguiente ejemplo se desactiva el 7 de abril y el 21 de abril:

```
my_dc.disabledRanges = [ new Date(2003, 3, 7), new Date(2003, 3, 21)  ];
```

DateChooser.displayedMonth

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.displayedMonth

Descripción

Propiedad; número que indica el mes que se visualiza. El número indica un elemento de la matriz `monthNames`, donde 0 indica el primer mes. El valor predeterminado es el mes correspondiente a la fecha actual.

Ejemplo

En el ejemplo siguiente, se define diciembre como el mes que se visualizará:

```
my_dc.displayedMonth = 11;
```

Véase también

DateChooser.displayedYear

DateChooser.displayedYear

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.displayedYear

Descripción

Propiedad; número de cuatro dígitos que indica el año que se visualiza. El valor predeterminado es el año actual.

Ejemplo

En el ejemplo siguiente, se define el año 2010 como el año que se visualizará:

```
my_dc.displayedYear = 2010;
```

Véase también

DateChooser.displayedMonth

DateChooser.firstDayOfWeek

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.firstDayOfWeek

Descripción

Propiedad; número que indica el día de la semana (de 0 a 6, donde 0 indica el primer elemento de la matriz `dayNames`) que se visualiza en la primera columna del componente DateChooser. Si se cambia esta propiedad se cambiará el orden de las columnas de día pero no se modificará el orden de la propiedad `dayNames`. El valor predeterminado es 0 (domingo).

Ejemplo

En el ejemplo siguiente se establece el lunes como primer día de la semana:

```
// Definir el primer día de la semana en lunes en el calendario.
my_dc.firstDayOfWeek = 1;

// Desactiva el día 0 (domingo). Aunque el lunes es ahora el primer día del
  componente DateChooser, el domingo sigue siendo el índice de matriz 0.
my_dc.disabledDays = [0];
```

Véase también

DateChooser.dayNames

DateChooser.monthNames

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.monthNames

Descripción

Propiedad; matriz de cadenas que indica los nombres de mes en la parte superior del componente DateChooser. El valor predeterminado es `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]`.

Ejemplo

En el ejemplo siguiente se definen los nombres de mes para la instancia `my_dc`:

```
my_dc.monthNames = ["Jan", "Feb","Mar","Apr", "May", "June","July", "Aug",
  "Sept","Oct", "Nov", "Dec"];
```

DateChooser.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  //...
}
dateChooserInstance.addEventListener("scroll", listenerObject)
```

Sintaxis 2:

```
on (scroll) {
  //...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic en un botón de mes.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*myDC*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto de evento del evento scroll tiene una propiedad adicional, `detail`, que puede tener uno de los valores siguientes: `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de DateChooser. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `myDC`, envía "_level0.myDC" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se hace clic en un botón de mes en una instancia de DateChooser denominada `my_dc`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `scroll` en el objeto detector. Dentro de la función hay una sentencia `trace()` que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo `evt_obj`, para generar un mensaje.

```
// Crear un objeto detector.
var dcListener:Object = new Object();
dcListener.scroll = function(evt_obj:Object) {
 trace(evt_obj.detail);
};

// Añadir objeto detector al selector de fecha.
my_dc.addEventListener("scroll", dcListener);
```

DateChooser.selectableRange

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.selectableRange

Descripción

Propiedad; establece una fecha única seleccionable o un rango de fechas seleccionables. El usuario no puede desplazarse fuera del rango seleccionable. El valor de esta propiedad es un objeto compuesto por dos objetos Date denominados `rangeStart` y `rangeEnd`. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas seleccionables. Si sólo se define `rangeStart`, se activarán todas las fechas posteriores a la señalada en `rangeStart`. Si sólo se define `rangeEnd`, se activarán todas las fechas anteriores a la señalada en `rangeEnd`. El valor predeterminado es `undefined`.

Si lo que se desea es activar un único día, puede utilizarse un único objeto Date como valor de `selectableRange`.

Especifique una fecha completa cuando defina fechas, por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, el objeto Date devuelve la fecha y hora actuales. Si no se especifica una hora, la hora devuelta es `00:00:00`.

El valor de `DateChooser.selectedDate` queda establecido como `undefined` si cae fuera del rango seleccionable.

Los valores de `DateChooser.displayedMonth` y `DateChooser.displayedYear` se establecen en el último mes más próximo del rango seleccionable si el mes actual cae fuera de dicho rango. Por ejemplo, si el mes que se muestra como actual es agosto (August) y el rango seleccionable comprende el periodo que va desde junio de 2003 a julio de 2003, el mes que se mostrará cambiará a julio de 2003 (July).

Ejemplo

En el ejemplo siguiente se define como rango seleccionable las fechas comprendidas entre el 7 de mayo y el 7 de junio, ambas inclusive:

```
my_dc.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de mayo y las fechas posteriores a ese día:

```
my_dc.selectableRange = {rangeStart: new Date(2005, 4, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de junio y las fechas anteriores a ese día:

```
my_dc.selectableRange = {rangeEnd: new Date(2005, 5, 7)};
```

En el ejemplo siguiente se define el 7 de junio como única fecha seleccionable:

```
my_dc.selectableRange = new Date(2005, 5, 7);
```

DateChooser.selectedDate

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateChooserInstance.selectedDate

Descripción

Propiedad; objeto Date que indica la fecha seleccionada si el valor se encuentra dentro del valor especificado en la propiedad `selectableRange`. El valor predeterminado es `undefined`.

No es posible establecer la propiedad `selectedDate` dentro de un rango desactivado, fuera de un rango seleccionable ni en un día que se ha desactivado. Si se establece esta propiedad en una de estas fechas, el valor es `undefined`.

Ejemplo

En el ejemplo siguiente se define el 7 de junio como fecha seleccionada:

```
my_dc.selectedDate = new Date(2005, 5, 7);
```

DateChooser.showToday

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
dateChooserInstance.showToday
```

Descripción

Propiedad; valor booleano que indica si debe resaltarse o no la fecha actual. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se desactiva el resaltado de la fecha correspondiente al día de hoy:

```
my_dc.showToday = false;
```

CAPÍTULO 15

Componente DateField (sólo en Flash Professional)

15

El componente DateField es un campo de texto no seleccionable que muestra la fecha con un icono de calendario situado a su derecha. Si no se ha seleccionado ninguna fecha, el campo de texto estará vacío y se mostrará el mes correspondiente al día de hoy en el selector de fechas. Cuando el usuario hace clic dentro del recuadro de delimitación del campo de fecha, aparece el selector de fechas con el mes correspondiente a la fecha seleccionada. Cuando el selector de fechas está abierto, el usuario puede usar los botones de desplazamiento para cambiar de mes y año y seleccionar una fecha. Cuando se selecciona una fecha, el selector de fechas se cierra y la selección se introduce en el campo de fecha.

NOTA

El campo de fecha se borra cuando un usuario selecciona la misma fecha dos veces. Para evitar que los usuarios anulen la selección de una fecha de forma accidental, consulte el ejemplo de `DateField.selectedDate` en la página 475.

La previsualización dinámica del componente DateField no refleja los valores indicados en el inspector de propiedades ni en el inspector de componentes durante la edición porque se trata de un componente emergente que no puede verse durante la edición.

Utilización del componente DateField (sólo en Flash Professional)

Utilice DateField siempre que desee que el usuario seleccione una fecha. Por ejemplo, puede utilizarse el componente DateField en un sistema de reservas de hotel en el que unas fechas determinadas puedan seleccionarse y otras queden desactivadas. También puede utilizarse el componente DateField en una aplicación que muestre, cuando el usuario elija una fecha, los eventos en curso, como por ejemplo actuaciones o reuniones.

Parámetros de DateField

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DateField en el inspector de propiedades o en el inspector de componentes:

dayNames establece el nombre de los días de la semana. El valor es una matriz y el valor predeterminado es `["S", "M", "T", "W", "T", "F", "S"]`.

disabledDays indica los días de la semana que están desactivados. Este parámetro es una matriz que puede tener hasta siete valores. El valor predeterminado es `[]` (matriz vacía).

firstDayOfWeek indica el día de la semana que se muestra en la primera columna del selector de fechas (0-6, con 0 como primer elemento de la matriz de `dayNames`). Esta propiedad cambia el orden de visualización de las columnas de los días.

El valor predeterminado es 0, que corresponde a "S" de Sunday (domingo).

monthNames establece el nombre de los meses que se muestran en la fila de encabezado del calendario. El valor es una matriz y el valor predeterminado es `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"]`.

showToday indica si debe resaltarse la fecha de hoy o no. El valor predeterminado es`true`.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes DateField utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase DateField (sólo en Flash Professional)" en la página 455.

Creación de aplicaciones con el componente DateField

El procedimiento siguiente explica cómo añadir un componente DateField a una aplicación durante la edición. En este ejemplo, el componente DateField permite al usuario elegir una fecha en un sistema de reservas de una compañía aérea. Todas las fechas anteriores a la de hoy deben quedar desactivadas. También debe desactivarse un rango de 15 días de diciembre, de manera que se cree un período de vacaciones que quede excluido. Además, los lunes hay algunos vuelos que no están disponibles, por lo que todos los lunes correspondientes a dichos vuelos deben quedar desactivados.

Para crear una aplicación con el componente DateField:

1. En el panel Componentes haga doble clic en el componente DateField para añadirlo al escenario.
2. En el inspector de propiedades, introduzca el nombre de instancia **flightCalendar**.
3. En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas seleccionables:

   ```
   flightCalendar.selectableRange = {rangeStart:new Date(2001, 9, 1),
     rangeEnd:new Date(2003, 11, 1)};
   ```

   Este código asigna un valor a la propiedad `selectableRange` de un objeto ActionScript que contenga dos objetos Date con los nombres de variable `rangeStart` y `rangeEnd`. De esta forma se definen los límites superior e inferior de un rango dentro del cual el usuario puede seleccionar una fecha.
4. En el panel Acciones introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer los intervalos de fechas desactivadas: una durante el mes de diciembre y otra para todas las fechas anteriores a la actual:

   ```
   flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
     rangeEnd: new Date(2003, 11, 31)}, {rangeEnd: new Date(2003, 6, 16)}];
   ```

5. En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para que los lunes queden desactivados:

   ```
   flightCalendar.disabledDays=[1];
   ```

6. Control > Probar película.

Para crear una instancia del componente DateField mediante código ActionScript:

1. Arrastre el componente DateField desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.
2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

   ```
   this.createClassObject(mx.controls.DateField, "my_df", 1);
   ```

   Este script utiliza el método `UIObject.createClassObject()` para crear la instancia de DateField.
3. Seleccione Control > Probar película.

Personalización del componente DateField (sólo en Flash Professional)

El componente DateField puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`). Al definir la anchura no se cambian las dimensiones del selector de fechas en el componente DateField. No obstante, se puede utilizar la propiedad `pullDown` para acceder al componente DateChooser y definir sus dimensiones.

Utilización de estilos con el componente DateField

Puede definir propiedades de estilo para cambiar el aspecto de una instancia de campo de fecha. Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente DateField admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Color de la iluminación cuando las fechas están seleccionadas y se pasa el puntero sobre ellas. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`
`backgroundColor`	Ambos	Color del fondo. El valor predeterminado es 0xEFEBEF (gris claro).
`borderColor`	Ambos	Color del borde. El valor predeterminado es 0x919999. La lista desplegable del componente DateField utiliza una línea sólida de un solo píxel como borde. Este borde no puede modificarse con estilos o aspectos.
`headerColor`	Ambos	El color de fondo para el encabezado de la lista desplegable. El color predeterminado es blanco.
`rollOverColor`	Ambos	Color de fondo de una fecha por la que se desplaza el puntero. El valor predeterminado es 0xE3FFD6 (verde brillante) con el tema Halo y 0xAAAAAA (gris claro) con el tema Sample.

Estilo	Tema	Descripción
`selectionColor`	Ambos	Color de fondo de la fecha seleccionada. El valor predeterminado es 0xCDFFC1 (verde claro) con el tema Halo y 0xEEEEEE (gris muy claro) con el tema Sample.
`todayColor`	Ambos	El color de fondo para la fecha de hoy. El valor predeterminado es 0x666666 (gris oscuro).
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

El componente DateField utiliza cuatro categorías de texto para mostrar el nombre del mes, los días de la semana, la fecha de hoy y las fechas habituales. Las propiedades de estilo de texto establecidas en el componente DateField controlan el texto de la fecha habitual y el texto mostrado en el estado contraído y proporcionan estilos predeterminados para los demás textos. Si se desea establecer estilos de texto para categorías de texto específicas, es necesario utilizar las siguientes declaraciones de estilo de clase.

Nombre de la declaración	Descripción
`HeaderDateText`	Nombre del mes.
`WeekDayStyle`	Días de la semana.
`TodayStyle`	Fecha de hoy.

En el siguiente ejemplo se muestra cómo establecer el nombre del mes y los días de la semana en rojo oscuro.

```
_global.styles.HeaderDateText.setStyle("color", 0x660000);
_global.styles.WeekDayStyle.setStyle("color", 0x660000);
```

Utilización de aspectos con el componente DateField

El componente DateField utiliza aspectos para representar los estados visuales del icono emergente, una instancia de RectBorder para el borde de la entrada de texto y una instancia de DateChooser para el icono emergente. Para aplicar aspectos al icono emergente durante la edición, modifique los símbolos de aspecto en la carpeta Flash UI Components 2/Themes/MMDefault/DateField Assets/States de la biblioteca de uno de los archivos FLA del tema. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*. Para más información sobre la aplicación de aspectos a las instancias de RectBorder y DateChooser, consulte "Clase RectBorder" en la página 1103 y "Utilización de aspectos con el componente DateChooser" en la página 430.

Además de los aspectos utilizados por los subcomponentes anteriormente mencionados, un componente DateField utiliza las siguientes propiedades de aspecto para aplicar dinámicamente un aspecto al icono emergente:

Propiedad	Descripción
`openDateUp`	Estado sin presionar del icono emergente.
`openDateDown`	Estado presionado del icono emergente.
`openDateOver`	Estado del icono emergente cuando se pasa el puntero sobre él.
`openDateDisabled`	Estado desactivado del icono emergente.

Para crear símbolos de clip de película para aspectos de DateField:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta DateField Assets a la biblioteca del documento.
4. Expanda la carpeta DateField Assets de la biblioteca del documento.
5. Asegúrese de que el símbolo DateFieldAssets está seleccionado para Exportar en primer fotograma.
6. Expanda la carpeta DateField Assets/States de la biblioteca del documento.
7. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo openIconUp.
8. Personalice el símbolo como desee.

   Por ejemplo, dibuje una flecha hacia abajo sobre la imagen del calendario.
9. Repita los pasos 7 y 8 en todos los símbolos que desee personalizar.

   Por ejemplo, dibuje una flecha hacia abajo sobre todos los símbolos.
10. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
11. Arrastre un componente DateField al escenario.
12. Seleccione Control > Probar película.

Clase DateField (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > ComboBase > DateField

Nombre de clase de ActionScript mx.controls.DateField

Las propiedades de la clase DateField permiten acceder a la fecha seleccionada y al mes y año mostrados. También se puede indicar el nombre de los días y los meses, las fechas desactivadas y seleccionables, establecer el primer día de la semana y decidir si debe resaltarse o no la fecha actual.

Si una propiedad de la clase DateField se define con código ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.DateField.version);
```

NOTA	El código `trace(myDateFieldInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase DateField

En la tabla siguiente se enumeran los métodos de la clase DateField.

Método	Descripción
`DateField.close()`	Cierra el subcomponente DateChooser emergente.
`DateField.open()`	Abre el subcomponente DateChooser emergente.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase DateField de la clase UIObject. Al llamar a estos métodos desde el objeto DateField, debe utilizarse la forma *`dateFieldInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.

Método	Descripción
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase DateField de la clase UIComponent. Al llamar a estos métodos desde el objeto DateField, debe utilizarse la forma *dateFieldInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase DateField

En la tabla siguiente se enumeran las propiedades de la clase DateField.

Propiedad	Descripción
DateField.dateFormatter	Función que da formato a la fecha que va a visualizarse en el campo de texto.
DateField.dayNames	Matriz que indica el nombre de los días de la semana.
DateField.disabledDays	Matriz que indica cuáles son los días de la semana que están desactivados.
DateField.disabledRanges	Rango de fechas desactivadas o fecha única desactivada.
DateField.displayedMonth	Número que indica qué elemento de la matriz monthNames se va a mostrar.
DateField.displayedYear	Número que indica el año que se mostrará.
DateField.firstDayOfWeek	Número que indica un elemento de la matriz dayNames que se mostrará en la primera columna del componente DateField.
DateField.monthNames	Matriz de cadenas que indica el nombre de los meses.
DateField.pullDown	Referencia al subcomponente DateChooser. Es una propiedad de sólo lectura.

Propiedad	Descripción
`DateField.selectableRange`	Fecha única seleccionable o rango de fechas seleccionables.
`DateField.selectedDate`	Objeto Date que indica la fecha seleccionada.
`DateField.showToday`	Valor booleano que indica si debe resaltarse o no la fecha actual.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase DateField de la clase UIObject. Al acceder a estas propiedades desde el objeto DateField, debe utilizarse la forma *`dateFieldInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase DateField de la clase UIComponent. Al acceder a estas propiedades desde el objeto DateField, debe utilizarse la forma *dateFieldInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase DateField

En la tabla siguiente se enumeran los eventos de la clase DateField.

Evento	Descripción
DateField.change	Se difunde cuando se selecciona una fecha.
DateField.close	Se difunde cuando se cierra el subcomponente DateChooser.
DateField.open	Se difunde cuando se abre el subcomponente DateChooser.
DateField.scroll	Se difunde cuando se hace clic en los botones de mes.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase DateField de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase DateField de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

DateField.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una fecha.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia del componente (*dateFieldInstance*) distribuye un evento (en este caso, change) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método EventDispatcher.addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador on() que debe asociarse directamente con una instancia de DateField. La palabra clave this, utilizada en un controlador on() asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas my_df, envía "_level0.my_df" al panel Salida:

```
on (change) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se cambia un campo de fecha denominado my_df. La primera línea de código crea un objeto detector denominado dfListener. La segunda línea define una función para el evento change en el objeto detector. Dentro de la función hay una sentencia trace() que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo evt_obj, para generar un mensaje. La propiedad target de un objeto de evento es el componente que ha generado el evento; en este ejemplo, my_df. Se accede a la propiedad DateField.selectedDate desde la propiedad target del objeto de evento. La última línea llama a EventDispatcher.addEventListener() desde my_df y pasa como parámetros el evento change y el objeto detector dfListener.

```
// Crear un objeto detector.
var dfListener:Object = new Object();
dfListener.change = function(evt_obj:Object){
 var thisDate:Date = evt_obj.target.selectedDate;
    trace("date selected: " + thisDate);
}

// Añadir objeto detector a DateField.
my_df.addEventListener("change", dfListener);
```

DateField.close()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.close()

Valor devuelto

Ninguno.

Descripción

Método; cierra el menú emergente.

Ejemplo

El código siguiente cierra el cuadro emergente del selector de fechas de la instancia de campo de fecha `my_df` cuando se hace clic en el botón `my_btn`:

```
// Crear un objeto detector.
var btnListener:Object = new Object();
btnListener.click = function() {
   my_df.close();
};

//Añadir un detector de Button
my_btn.addEventListener("click", btnListener);
```

DateField.close

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("close", listenerObject);
```

Sintaxis 2:

```
on (close) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el subcomponente DateChooser se cierra como consecuencia de que un usuario ha hecho clic en un área fuera del icono o ha seleccionado una fecha.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*dateFieldInstance*) distribuye un evento (en este caso, `close`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de DateField. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `my_df`, envía "_level0.my_df" al panel Salida:

```
on (close) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se cierra el selector de fechas de `my_df`. La primera línea de código crea un objeto detector denominado `dfListener`. La segunda línea define una función para el evento `close` en el objeto detector. Dentro de la función hay una sentencia `trace()` que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo `evt_obj`, para generar un mensaje. La propiedad `target` de un objeto de evento es el componente que ha generado el evento; en este ejemplo, `my_df`. Se accede a la propiedad `selectedDate` desde la propiedad `target` del objeto de evento. La última línea llama a `EventDispatcher.addEventListener()` desde `my_df` y pasa como parámetros el evento `close` y el objeto detector `dfListener`.

```
// Crear un objeto detector.
var dfListener:Object = new Object();
dfListener.close = function(evt_obj:Object){
  trace("PullDown Closed" + evt_obj.target.selectedDate);
}
// Añadir objeto detector a DateField.
my_df.addEventListener("close", dfListener);
```

DateField.dateFormatter

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateFieldInstance``.dateFormatter`

Descripción

Propiedad; función que da formato a la fecha que va a visualizarse en el campo de texto. La función debe recibir un objeto Date como parámetro y devolver una cadena en el formato que va a visualizarse.

Ejemplo

En el ejemplo siguiente se define la función para que devuelva el formato de fecha que va a visualizarse:

```
my_df.dateFormatter = function(d:Date){
  return d.getFullYear()+"/ "+(d.getMonth()+1)+"/ "+d.getDate();
};
```

DateField.dayNames

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.dayNames

Descripción

Propiedad; matriz que contiene el nombre de los días de la semana. El domingo es el primer día (en la posición de índice 0) y el resto de los nombres de día siguen en el orden habitual. El valor predeterminado es `["S", "M", "T", "W", "T", "F", "S"]`.

Ejemplo

En el ejemplo siguiente el valor del quinto día de la semana, jueves (Thursday), cambia de "T" a "R":

```
my_df.dayNames[4] = "R";
```

En el siguiente ejemplo se cambia el valor de todos los días, según corresponda:

```
my_df.dayNames = new Array("Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");
```

DateField.disabledDays

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.disabledDays

Descripción

Propiedad; matriz que indica cuáles son los días de la semana que están desactivados. Todas las fechas del mes que coincidan con el día especificado quedarán desactivadas. Los valores de los elementos de esta matriz pueden estar comprendidos entre el 0, para el domingo, y el 6, para el sábado. El valor predeterminado es `[]` (matriz vacía).

Ejemplo

En el ejemplo siguiente se desactivan los domingos y los sábados de forma que los usuarios sólo puedan seleccionar el resto de los días:

```
my_df.disabledDays = [0, 6];
```

DateField.disabledRanges

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.disabledRanges

Descripción

Propiedad; desactiva un único día o un rango de días. Esta propiedad es una matriz de objetos. Cada objeto de la matriz debe ser un objeto Date, que especifique una fecha única para desactivarla, o bien un objeto que contenga las propiedades `rangeStart` y `rangeEnd` (o al menos una de ellas), cuyos valores deben ser un objeto Date. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas. Si se omite alguna de dichas propiedades, el rango quedará abierto por la fecha correspondiente.

El valor predeterminado de `disabledRanges` es `undefined`.

Especifique una fecha completa cuando defina fechas para la propiedad `disabledRanges`, por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, la hora devuelta es `00:00:00`.

Ejemplo

En el ejemplo siguiente se define una matriz con los objetos Date `rangeStart` y `rangeEnd`, que desactivan las fechas comprendidas entre el 7 de mayo y el 7 de junio:

```
my_df.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

En el ejemplo siguiente se desactivan todas las fechas posteriores al 7 de noviembre:

```
my_df.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

En el ejemplo siguiente se desactivan todas las fechas anteriores al 7 de octubre:

```
my_df.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

En el ejemplo siguiente se desactiva únicamente el 7 de diciembre:

```
my_df.disabledRanges = [ new Date(2003, 11, 7) ];
```

En el ejemplo se siguiente desactiva el 7 de diciembre y el 20 de diciembre:

```
my_df.disabledRanges = [ new Date(2003, 11, 7), new Date(2003, 11, 20)];
```

DateField.displayedMonth

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.displayedMonth

Descripción

Propiedad; número que indica el mes que se visualiza. El número indica un elemento de la matriz `monthNames`, donde 0 indica el primer mes. El valor predeterminado es el mes correspondiente a la fecha actual.

Ejemplo

En el ejemplo siguiente, se define diciembre como el mes que se visualizará:

```
my_df.displayedMonth = 11;
```

Véase también

DateField.displayedYear

DateField.displayedYear

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.displayedYear

Descripción

Propiedad; número que indica el año que se visualiza. El valor predeterminado es el año actual.

Ejemplo

En el ejemplo siguiente, se define el año 2010 como el año que se visualizará:

```
my_df.displayedYear = 2010;
```

Véase también

`DateField.displayedMonth`

DateField.firstDayOfWeek

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateFieldInstance``.firstDayOfWeek`

Descripción

Propiedad; número que indica el día de la semana (de 0 a 6, donde 0 indica el primer elemento de la matriz `dayNames`) que se visualiza en la primera columna del componente DateField. Si se cambia esta propiedad se cambiará el orden de las columnas de día pero no se modificará el orden de la propiedad `dayNames`. El valor predeterminado es 0 (domingo).

Ejemplo

En el ejemplo siguiente se establece el lunes como primer día de la semana:

```
my_df.firstDayOfWeek = 1;
```

Véase también

`DateField.dayNames`

DateField.monthNames

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.monthNames

Descripción

Propiedad; matriz de cadenas que indica los nombres de mes en la parte superior del componente DateField. El valor predeterminado es `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]`.

Ejemplo

En el ejemplo siguiente se definen los nombres de mes para la instancia `my_df`:

```
my_df.monthNames = ["Jan", "Feb","Mar","Apr", "May", "June","July", "Aug",
  "Sept","Oct", "Nov", "Dec"];
```

DateField.open()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.open()

Valor devuelto

Ninguno.

Descripción

Método; abre el subcomponente DateChooser emergente.

Ejemplo

El código siguiente abre el cuadro emergente del selector de fechas de la instancia de campo de fecha `my_df` cuando se hace clic en el botón `my_btn`:

```
// Crear un objeto detector.
var btnListener:Object = new Object();
btnListener.click = function() {
   my_df.open();
};

//Añadir un detector de Button
my_btn.addEventListener("click", btnListener);
```

DateField.open

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.open = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("open", listenerObject);
```

Sintaxis 2:

```
on (open) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un subcomponente DateChooser se abre como consecuencia de que un usuario ha hecho clic en el icono.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*dateFieldInstance*) distribuye un evento (en este caso, open) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método EventDispatcher.addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador on() que debe asociarse directamente con una instancia de DateField. La palabra clave this, utilizada en un controlador on() asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas my_df, envía "_level0.my_df" al panel Salida:

```
on (open) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se abre un campo de fecha denominado my_df. El mensaje envía el número de índice para el mes actual. La primera línea de código crea un objeto detector denominado dfListener. La segunda línea define una función para el evento open en el objeto detector. Dentro de la función hay una sentencia trace() que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo evt_obj, para generar un mensaje. La propiedad target de un objeto de evento es el componente que ha generado el evento; en este ejemplo, my_df. Se accede a la propiedad DateField.selectedDate desde la propiedad target del objeto de evento. La última línea llama a EventDispatcher.addEventListener() desde my_df y pasa como parámetros el evento close y el objeto detector dfListener.

```
// Crear un objeto detector.
var dfListener:Object = new Object();
dfListener.open = function(evt_obj:Object){
  trace("PullDown Opened" + evt_obj.target.displayedMonth);
}
// Añadir objeto detector a DateField.
my_df.addEventListener("open", dfListener);
```

DateField.pullDown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

dateFieldInstance.pullDown

Descripción

Propiedad (sólo lectura); referencia al componente DateChooser contenido en el componente DateField. Cuando un usuario hace clic en el componente DateField se crea una instancia del subcomponente DateChooser. No obstante, si se hace referencia a la propiedad `pullDown` antes de que el usuario haga clic en el componente, se crea una instancia de DateChooser y después se oculta.

Ejemplo

En el ejemplo siguiente se establece el valor `false` para la visibilidad del subcomponente DateChooser y, a continuación, se establece el tamaño de dicho subcomponente: 300 píxeles de altura y 300 píxeles de anchura.

```
my_df.pullDown._visible = false;
my_df.pullDown.setSize(300, 300);
```

DateField.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("scroll", listenerObject);
```

Sintaxis 2:

```
on (scroll) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic en un botón de mes.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia del componente (*dateFieldInstance*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto de evento del evento scroll tiene una propiedad adicional, `detail`, que puede tener uno de los valores siguientes: `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de DateField. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `my_df`, envía "_level0.my_df" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se hace clic en un botón de mes en una instancia de DateField denominada `my_df`. La primera línea de código crea un objeto detector denominado `dfListener`. La segunda línea define una función para el evento `scroll` en el objeto detector. Dentro de la función hay una sentencia `trace()` que utiliza el objeto de evento pasado automáticamente a la función, en este ejemplo `evt_obj`, para generar un mensaje. La propiedad `target` de un objeto de evento es el componente que generó el evento, en este ejemplo, `my_df`. La última línea lama a `EventDispatcher.addEventListener()` desde `my_df` y pasa como parámetros el evento `scroll` y el objeto detector `dfListener`.

```
// Crear un objeto detector.
var dfListener:Object = new Object();
dfListener.scroll = function(evt_obj:Object) {
 trace(evt_obj.detail);
};

// Añadir objeto detector a DateField.
my_df.addEventListener("scroll", dfListener);
```

DateField.selectableRange

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateFieldInstance``.selectableRange`

Descripción

Propiedad; establece una fecha única seleccionable o un rango de fechas seleccionables. El valor de esta propiedad es un objeto compuesto por dos objetos Date denominados `rangeStart` y `rangeEnd`. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas elegible. Si sólo se define `rangeStart`, se activarán todas las fechas posteriores a la señalada en `rangeStart`. Si sólo se define `rangeEnd`, se activarán todas las fechas anteriores a la señalada en `rangeEnd`. El valor predeterminado es `undefined`.

Si lo que se desea es activar un único día, puede utilizarse un único objeto Date como valor de `selectableRange`.

Especifique una fecha completa cuando defina fechas, por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, la hora devuelta es `00:00:00`.

El valor de `DateField.selectedDate` queda establecido como `undefined` si cae fuera del rango seleccionable.

Los valores de `DateField.displayedMonth` y `DateField.displayedYear` se establecen en el último mes más próximo del rango seleccionable si el mes actual cae fuera de dicho rango. Por ejemplo, si el mes que se muestra como actual es agosto (August) y el rango seleccionable comprende el periodo que va desde junio de 2003 a julio de 2003, el mes que se mostrará cambiará a julio de 2003 (July).

Ejemplo

En el ejemplo siguiente se define como rango seleccionable las fechas comprendidas entre el 7 de mayo y el 7 de junio, ambas inclusive:

```
my_df.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de mayo y las fechas posteriores a ese día:

```
my_df.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de junio y las fechas anteriores a ese día:

```
my_df.selectableRange = {rangeEnd: new Date(2003, 5, 7)};
```

En el ejemplo siguiente se define el 7 de junio como única fecha seleccionable:

```
my_df.selectableRange = new Date(2003, 5, 7);
```

DateField.selectedDate

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateFieldInstance``.selectedDate`

Descripción

Propiedad; objeto Date que indica la fecha seleccionada si el valor se encuentra dentro del valor especificado en la propiedad `selectableRange`. El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se define el 7 de junio como fecha seleccionada:

```
my_df.selectedDate = new Date(2003, 5, 7);
```

En el ejemplo siguiente se utiliza una instancia de DateField denominada `my_df` en el escenario para mostrar cómo desactivar una fecha ya seleccionada (de lo contrario, el usuario puede hacer clic en ella de nuevo para borrar la entrada de campo de fecha).

```
function dfListener(evt_obj:Object):Void {
  my_df.disabledRanges = [my_df.selectedDate];
}
my_df.addEventListener("change", dfListener);
```

DateField.showToday

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`dateFieldInstance``.showToday`

Descripción

Propiedad; valor booleano que indica si debe resaltarse o no la fecha actual. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se desactiva el resaltado de la fecha correspondiente al día de hoy:

```
my_df.showToday = false;
```

CAPÍTULO 16

Clase Delegate

16

Herencia Objeto > Delegate

Nombre de clase de ActionScript mx.utils.Delegate

La clase Delegate permite ejecutar una función en un ámbito específico. Esta clase se proporciona para que se pueda distribuir el mismo evento en dos funciones diferentes (véase "Delegación de eventos a funciones" en *Utilización de componentes*) y para que se pueda llamar a las funciones dentro del ámbito de la clase que las contiene.

Cuando se pasa una función como un parámetro a `EventDispatcher.addEventListener()`, la función se invoca en el ámbito de la instancia del componente difusor, no del objeto en el que está declarada (véase "Delegación del ámbito de una función" en *Utilización de componentes*). Se puede llamar a `Delegate.create()` para llamar a la función del ámbito del objeto que declara.

Resumen de métodos de la clase Delegate

En la tabla siguiente se muestra el método de la clase Delegate.

Método	Descripción
`Delegate.create()`	Método estático que permite ejecutar una función en un ámbito específico.

Delegate.create()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
Delegate.create(scopeObject, function)
```

Parámetros

scopeObject Referencia a un objeto. Éste es el ámbito en el que hay que ejecutar la función.

function Referencia a una función.

Descripción

Método (estático); permite delegar eventos a funciones y ámbitos específicos. Utilice la siguiente sintaxis:

```
import mx.utils.Delegate;
compInstance.addEventListener("eventName", Delegate.create(scopeObject,
  function));
```

El parámetro *scopeObject* especifica el ámbito en el que se llama a una función especificada.

Ejemplo

Para ver ejemplos de `Delegate.create()`, consulte "Delegación de eventos" en *Utilización de componentes*.

Véase también

`EventDispatcher.addEventListener()`

CAPÍTULO 17

Clase DeltaItem (sólo en Flash Professional)

17

Nombre de clase de ActionScript mx.data.components.datasetclasses.DeltaItem

La clase DeltaItem proporciona información acerca de una operación individual llevada a cabo en un objeto de transferencia. Indica si un cambio se realizó directamente en una propiedad del objeto de transferencia o mediante una llamada a método. También proporciona el estado original de las propiedades de un objeto de transferencia. Por ejemplo, si el origen del paquete delta es un juego de datos, el objeto DeltaItem contendrá información sobre cualquier campo que se haya editado.

Además de lo anterior, un objeto DeltaItem puede contener información de respuesta del servidor como, por ejemplo, el valor actual y un mensaje.

Se puede utilizar la clase DeltaItem para acceder a los cambios del paquete delta. Para acceder a estos cambios, utilice `DeltaPacket.getIterator()`, que devuelve un repetidor de deltas. Cada delta contiene 0 o más instancias de DeltaItem a las que se puede acceder mediante `Delta.getItemByName()` o `Delta.getChangeList()`.

Resumen de propiedades de la clase DeltaItem

En la siguiente tabla se enumeran las propiedades de la clase DeltaItem.

Propiedad	Descripción
`DeltaItem.argList`	Si se realiza un cambio durante una llamada a método, es la matriz de valores que se pasaron al método. Es una propiedad de sólo lectura.
`DeltaItem.curValue`	Si se realiza un cambio en una propiedad, es el valor actual de servidor de la propiedad. Es una propiedad de sólo lectura.
`DeltaItem.delta`	Delta asociado para el objeto DeltaItem. Es una propiedad de sólo lectura.
`DeltaItem.kind`	Tipo de cambio.
`DeltaItem.message`	Mensaje de servidor asociado al objeto DeltaItem.

Propiedad	Descripción
`DeltaItem.name`	Nombre de la propiedad o método que ha cambiado. Es una propiedad de sólo lectura.
`DeltaItem.newValue`	Si se realiza un cambio en una propiedad, es el nuevo valor de la propiedad. Es una propiedad de sólo lectura.
`DeltaItem.oldValue`	Si se realiza un cambio en una propiedad, es el valor antiguo de la propiedad. Es una propiedad de sólo lectura.

DeltaItem.argList

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.argList

Descripción

Propiedad (sólo lectura); matriz de valores pasados al método de cambio. Esta propiedad sólo se aplica si el tipo de cambio es `DeltaItem.Method`.

DeltaItem.curValue

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.curValue

Descripción

Propiedad (sólo lectura); objeto que contiene el valor de la propiedad actual en la copia del servidor del objeto de transferencia. Esta propiedad se aplica sólo si el tipo de cambio es `DeltaItem.Property` y la propiedad es relevante sólo en un delta que haya devuelto el servidor y si se está aplicando a un juego de datos que debe resolver el usuario.

DeltaItem.delta

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.delta

Descripción

Propiedad (sólo lectura); delta asociado al objeto DeltaItem. Cuando se crea un objeto DeltaItem, éste se asocia a un delta y se añade a sí mismo a la lista de cambios del delta. Esta propiedad proporciona una referencia para el delta al que pertenece este elemento.

DeltaItem.kind

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.kind

Descripción

Propiedad; número que indica el tipo de cambio. Utilice las siguientes constantes para evaluar esta propiedad:

- `DeltaItem.Property` El cambio se realizó en una propiedad del objeto de transferencia.
- `DeltaItem.Method` El cambio se realizó mediante una llamada a método en el objeto de transferencia.

DeltaItem.message

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.message

Descripción

Propiedad; cadena que contiene un mensaje de servidor asociado a este objeto DeltaItem. Puede ser cualquier mensaje relativo al intento de cambio de llamada a método o de propiedad en el paquete delta. Este mensaje suele ser relevante sólo en un delta que haya devuelto el servidor y que se está aplicando a DataSet para resolverlo.

DeltaItem.name

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.name

Descripción

Propiedad (sólo lectura); cadena que contiene el nombre de la propiedad cambiada (si el tipo de cambio es `DeltaItem.Property`) o el nombre del método que realizó el cambio (si el tipo de cambio es `DeltaItem.Method`).

DeltaItem.newValue

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.newValue

Descripción

Propiedad (sólo lectura); objeto que contiene el valor nuevo de la propiedad. Esta propiedad sólo se aplica si el tipo de cambio es `DeltaItem.Property`.

DeltaItem.oldValue

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaitem.oldValue

Descripción

Propiedad (sólo lectura); objeto que contiene el valor antiguo de la propiedad. Esta propiedad sólo se aplica si el tipo de cambio es `DeltaItem.Property`.

CAPÍTULO 18

Interfaz Delta (sólo en Flash Professional)

18

Nombre de interfaz de ActionScript mx.data.components.datasetclasses.Delta

La interfaz Delta proporciona acceso al objeto de transferencia, a la colección y a los cambios en el objeto de transferencia. Con esta interfaz se puede acceder a valores nuevos y anteriores de un objeto de transferencia. Por ejemplo, si el paquete delta se obtuvo de un juego de datos, cada delta representaría una fila añadida, editada o eliminada.

La interfaz Delta también proporciona acceso a los mensajes devueltos por el proceso de servidor asociado. Para más información sobre las interacciones cliente-servidor, consulte "Componente RDBMSResolver (sólo en Flash Professional)" en la página 1087.

La interfaz Delta puede utilizarse para examinar el paquete delta antes de enviar los cambios al servidor y para revisar los mensajes devueltos por el procesamiento del servidor.

Resumen de métodos de la interfaz Delta

En la siguiente tabla se enumeran los métodos de la interfaz Delta.

Método	Descripción
`Delta.addDeltaItem()`	Añade la instancia de DeltaItem especificada.
`Delta.getChangeList()`	Devuelve una matriz de cambios realizados en el elemento actual.
`Delta.getDeltaPacket()`	Devuelve el paquete delta que contiene el delta.
`Delta.getId()`	Devuelve el ID exclusivo del elemento actual de la colección DeltaPacket.
`Delta.getItemByName()`	Devuelve el objeto DeltaItem especificado.
`Delta.getMessage()`	Devuelve el mensaje asociado al elemento actual.
`Delta.getOperation()`	Devuelve la operación que se llevó a cabo en el elemento actual en la colección original.
`Delta.getSource()`	Devuelve el objeto de transferencia en el que se han realizado cambios.

Delta.addDeltaItem()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

delta.addDeltaItem(*deltaitem*)

Parámetros

deltaitem Instancia DeltaItem que va a añadirse a este delta.

Valor devuelto

Ninguno.

Descripción

Método; añade la instancia DeltaItem especificada. Si la instancia DeltaItem existe ya, este método la reemplaza.

Ejemplo

En el siguiente ejemplo se llama al método `addDeltaItem()`:

```
//...
var d:Delta = new DeltaImpl("ID1345678", curItem, DeltaPacketConsts.Added,
  "", false);
d.addDeltaItem(new DeltaItem(DeltaItem.Property, "ID", {oldValue:15,
  newValue:16}));
//...
```

Delta.getChangeList()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

delta.getChangeList()

Parámetros

Ninguno.

Valor devuelto

Matriz de instancias de DeltaItem asociadas.

Descripción

Método; devuelve una matriz de instancias de DeltaItem asociadas. Todas las instancias de DeltaItem de la matriz describen un cambio realizado en el elemento.

Ejemplo

En el siguiente ejemplo se llama al método `getChangeList()`:

```
//...
case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
  // dpDelta es una variable de tipo Delta.
  var changes:Array = dpDelta.getChangeList();
  for(var i:Number = 0; i<changes.length; i++) {
    // getChangeMessage es un método definido por el usuario.
    changeMsg = _parent.getChangeMessage(changes[i]);
    trace(changeMsg);
  }
//...
```

Delta.getDeltaPacket()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

delta`.getDeltaPacket()`

Parámetros

Ninguno.

Valor devuelto

El paquete delta que contiene este delta.

Descripción

Método; devuelve el paquete delta que contiene este delta. Este método permite escribir código para controlar los paquetes delta de forma genérica en el nivel delta.

Ejemplo

En el siguiente ejemplo se utiliza el método `getDeltaPacket()` para acceder al origen de datos del paquete delta:

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  trace("DeltaPacket source is: " + dpDelta.getDeltaPacket().getSource());
}
```

Delta.getId()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`delta``.getId()`

Parámetros

Ninguno.

Valor devuelto

Objeto; devuelve el ID exclusivo de este elemento de la colección DeltaPacket.

Descripción

Método; devuelve el identificador exclusivo para este elemento dentro de la colección DeltaPacket. Utilice este ID en el componente de origen para que el paquete delta reciba actualizaciones y realice cambios en los elementos con los que se generó el paquete delta. Por ejemplo, si suponemos que el componente DataSet envía actualizaciones a un servidor y que el servidor devuelve valores de campo clave nuevos, este método permite al componente DataSet examinar el paquete delta resultante, encontrar el objeto de transferencia original y actualizarlo de forma correspondiente.

Ejemplo

En el siguiente ejemplo se llama al método `getId()`:

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  trace("id ["+dpDelta.getId()+"]");
}
```

Delta.getItemByName()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`delta``.getItemByName(`*`name`*`)`

Parámetros

`name` Cadena que especifica el nombre de la propiedad o método para el objeto DeltaItem asociado.

Valor devuelto

Objeto DeltaItem especificado por *`name`*. Si no se encuentra ningún objeto DeltaItem que coincida con *`name`*, este método devuelve `null`.

Descripción

Método; devuelve el objeto DeltaItem especificado por *`name`*. Cuando name requiere llamadas a un método o cambios de valores de propiedad en un objeto de transferencia, este método proporciona el acceso más eficiente.

Ejemplo

En el siguiente ejemplo se llama al método `getItemByName()`:

```
private function buildFieldTag(deltaObj:Delta, field:Object,
  isKey:Boolean):String {
  var chgItem:DeltaItem = deltaObj.getItemByName(field.name);
  var result:String= "<field name=\"" + field.name + "\" type=\"" +
  field.type.name + "\"";
  var oldValue:String;
  var newValue:String;
  if (deltaObj.getOperation() != DeltaPacketConsts.Added) {
    oldValue = (chgItem != null ? (chgItem.oldValue != null ?
  encodeFieldValue(field.name, chgItem.oldValue) : __nullValue) :
  encodeFieldValue(field.name, deltaObj.getSource()[field.name]));
    newValue = (chgItem.newValue != null ? encodeFieldValue(field.name,
  chgItem.newValue) : __nullValue);
    result+= " oldValue=\""  + oldValue + "\"";
    result+= chgItem != null ? " newValue=\"" + newValue + "\"" : "";
    result+= " key=\"" + isKey.toString() + "\" />";
  }
  else {
    result+= " newValue=\"" +encodeFieldValue(field.name,
  deltaObj.getSource()[field.name]) + "\"";
    result+= " key=\"" + isKey.toString() + "\" />";
  }
  return result;
}
```

Delta.getMessage()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`delta``.getMessage()`

Parámetros

Ninguno.

Valor devuelto

Cadena; devuelve el mensaje asociado a *`delta`*.

Descripción

Método; devuelve el mensaje asociado para este delta. Normalmente, este mensaje sólo se llena si el paquete delta se ha devuelto desde un servidor como respuesta a intentos de actualización. Para más información, consulte "Componente RDBMSResolver (sólo en Flash Professional)" en la página 1087.

Ejemplo

En el siguiente ejemplo se llama al método `getMessage()`:

```
//...
var dpi:Iterator = dp.getIterator();
var d:Delta;
while(dpi.hasNext()) {
  d= dpi.next();
  trace(d.getMessage());
}
//...
```

Delta.getOperation()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`delta``.getOperation()`

Parámetros

Ninguno.

Valor devuelto

Número; devuelve la operación que se llevó a cabo en el elemento dentro de la colección original.

Descripción

Método; devuelve la operación que se llevó a cabo en este elemento dentro de la colección original. Los valores válidos para éste son `DeltaPacketConsts.Added`, `DeltaPacketConsts.Removed` y `DeltaPacketConsts.Modified`.

Es necesario importar mx.data.components.datasetclasses.DeltaPacketConsts o completar cada constante.

Ejemplo

En el siguiente ejemplo se llama al método `getOperation()`:

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
  trace("DeltaPacket source is: " + dpDelta.getDeltaPacket().getSource());
  switch(op) {
    case mx.data.components.datasetclasses.DeltaPacketConsts.Added:
      trace("***In case DeltaPacketConsts.Added ***");
    case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
      trace("***In case DeltaPacketConsts.Modified ***");
  }
}
```

Delta.getSource()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`delta``.getSource()`

Parámetros

Ninguno.

Valor devuelto

Objeto de transferencia en el que se han realizado cambios.

Descripción

Método; devuelve el objeto de transferencia en el que se han realizado cambios.

Ejemplo

En el siguiente ejemplo se llama al método `getSource()`:

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
  switch(op) {
    case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
    // los valores originales son
    trace("Unmodified source is: ");
    var src = dpDelta.getDeltaPacket().getSource();
    for(var i in src){
      if(typeof(src[i]) != "function"){
        trace(i+"="+src[i]);
      }
    }
  }
}
```

CAPÍTULO 19

Interfaz DeltaPacket (sólo en Flash Professional)

19

Nombre de interfaz de ActionScript mx.data.components.datasetclasses.DeltaPacket

La interfaz DeltaPacket se obtiene con la propiedad `deltaPacket` del componente DataSet, que forma parte de las funciones de administración de datos de Flash MX Professional 2004. (Para más información, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.) Normalmente, es el componente Resolver el que utiliza internamente el paquete delta. La interfaz DeltaPacket y la interfaz Delta y la clase DeltaItem relacionadas permiten administrar los cambios realizados en los datos. Estos componentes no se muestran visualmente en tiempo de ejecución.

Un paquete delta es un juego de instrucciones optimizado que describe todos los cambios realizados en los datos dentro de un juego de datos. Cuando se llama al método `DataSet.applyUpdates()`, el componente DataSet llena la propiedad `DataSet.deltaPacket`. Normalmente, esta propiedad está conectada (mediante vinculación de datos) a un componente Resolver como RDBMSResolver. El Resolver convierte el paquete delta en un paquete de actualización en el formato apropiado.

NOTA	A menos que se esté escribiendo un Resolver personalizado, es poco probable que haga falta conocer o escribir código que tenga acceso a los métodos o propiedades de un paquete delta.

Un paquete delta contiene uno o varios deltas (véase "Interfaz Delta (sólo en Flash Professional)" en la página 485) y cada delta contiene cero o más elementos delta (véase "Clase DeltaItem (sólo en Flash Professional)" en la página 479).

Resumen de métodos de la interfaz DeltaPacket

En la siguiente tabla se enumeran los métodos de la interfaz DeltaPacket.

Método	Descripción
`DeltaPacket.getConfigInfo()`	Devuelve información de configuración específica de la implementación de la interfaz DeltaPacket.
`DeltaPacket.getIterator()`	Devuelve el repetidor del paquete delta que se repite en la lista de deltas del paquete delta.
`DeltaPacket.getSource()`	Devuelve el origen del paquete delta. Es el componente que ha expuesto este paquete delta.
`DeltaPacket.getTimestamp()`	Devuelve la fecha y la hora en la que se creó una instancia del paquete delta.
`DeltaPacket.getTransactionId()`	Devuelve el ID de transacción para el paquete delta.
`DeltaPacket.logChanges()`	Indica si el usuario del paquete delta debe registrar los cambios que éste especifica.

DeltaPacket.getConfigInfo()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaPacket.getConfigInfo(*info*)

Properties

info Objeto; contiene información específica para la implementación.

Valor devuelto

Un objeto que contiene información necesaria para la implementación de DeltaPacket específica.

Descripción

Método; devuelve información de configuración específica de la implementación de la interfaz DeltaPacket. Este método permite implementar la interfaz DeltaPacket para acceder a la información personalizada.

Ejemplo

En el siguiente ejemplo se llama al método `getConfigInfo()`:

```
// ...
new DeltaPacketImpl(source, getTransactionId(), null, logChanges(),
  getConfigInfo());
// ...
```

DeltaPacket.getIterator()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`deltaPacket``.getIterator()`

Parámetros

Ninguno.

Valor devuelto

Una interfaz para el repetidor de la colección DeltaPacket que se repite en la lista de deltas del paquete delta.

Descripción

Método; devuelve el repetidor de la colección DeltaPacket. Esto proporciona un mecanismo de reproducción indefinida de los cambios en el paquete delta. Para obtener una descripción completa de esta interfaz, consulte "Interfaz Iterator (sólo en Flash Professional)" en la página 777.

Ejemplo

En el siguiente ejemplo se utiliza el método `getIterator()` para acceder al repetidor de los deltas en un paquete delta y se utiliza una sentencia `while` para reproducir indefinidamente los deltas:

```
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
trace("*** Test deltapacket. Trans ID is: " + deltapkt.getTransactionId() +
  " ***");
var OPS:Array = new Array("added", "removed", "modified");
var dpCursor:Iterator = deltapkt.getIterator();
var dpDelta:Delta;
var op:Number;
var changeMsg:String;
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
}
```

DeltaPacket.getSource()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`deltaPacket``.getSource()`

Parámetros

Ninguno.

Valor devuelto

Un objeto; el origen de la colección DeltaPacket. Este objeto es, normalmente, un descendiente de MovieClip, aunque no es necesario. Por ejemplo, si el origen es un juego de datos, este objeto puede ser `_level0.myDataSet`.

Descripción

Método; devuelve el origen de la colección DeltaPacket.

Ejemplo

En el siguiente ejemplo se llama al método `getSource()`:

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
var dpSourceText:String = "Source: " + deltapkt.getSource();
trace(dpSourceText);
// ...
```

DeltaPacket.getTimestamp()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`deltaPacket``.getTimestamp()`

Parámetros

Ninguno.

Valor devuelto

Un objeto Date que contiene la fecha y la hora en la que se creó el paquete delta.

Descripción

Método; devuelve la fecha y la hora en la que se creó el paquete delta.

Ejemplo

En el siguiente ejemplo se llama al método `getTimestamp()`:

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
var dpTime:String = " Time: " + deltapkt.getTimestamp();
trace(dpTime);
// ...
```

DeltaPacket.getTransactionId()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaPacket.getTransactionId()

Parámetros

Ninguno.

Valor devuelto

Una cadena; ID de transacción exclusivo para una sola transacción que agrupa paquetes delta.

Descripción

Método; devuelve el ID de transacción para el paquete delta. Este identificador exclusivo se utiliza para agrupar una transacción de envío/recepción de un paquete delta. El juego de datos lo utiliza para determinar si el paquete delta forma parte de la misma transacción que originó con la llamada a `DataSet.applyUpdates()`.

Ejemplo

En el siguiente ejemplo se llama al método `getTransactionId()`:

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
trace("*** Trans ID is: " + deltapkt.getTransactionId() + " ***");
// ...
```

DeltaPacket.logChanges()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

deltaPacket.logChanges()

Parámetros

Ninguno.

Valor devuelto

Un valor booleano; `true` si el usuario del paquete delta tuviera que registrar los cambios encontrados en el paquete delta.

Descripción

Método; devuelve `true` si el usuario de este paquete delta tuviera que registrar los cambios que éste especifica. Este valor se utiliza principalmente para comunicar cambios entre los juegos de datos mediante objetos compartidos o desde un servidor a un juego de datos local. En ambos casos, el juego de datos no debería registrar los cambios especificados.

Ejemplo

En el siguiente ejemplo se llama al método `logChanges()`:

```
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
if(deltapkt.logChanges()) {
  trace("*** We need to log changes. ***");
}
else {
  trace("*** We do not need to log changes");
}
```

CAPÍTULO 20

Clase DepthManager

20

Nombre de clase de ActionScript mx.managers.DepthManager

La clase DepthManager le permite administrar las asignaciones de profundidad relativa de cualquier componente o clip de película, incluido `_root`. También permite administrar profundidades reservadas en un clip especial de profundidad máxima situado en `_root` de servicios del sistema, como el puntero y las sugerencias.

En general, Depth Manager administra los componentes de forma automática utilizando su propio algoritmo de ordenación aleatoria. No es necesario utilizar sus interfaces API, a menos que se trate de un desarrollador de Flash avanzado.

NOTA

Para utilizar la clase DepthManager para instancias de clip de película, necesita tener un componente en la biblioteca o en el escenario y utilizar "import mx.managers.DepthManager" al principio del código ActionScript.

Los métodos siguientes componen la interfaz API de clasificación de profundidad relativa:

- `DepthManager.createChildAtDepth()`
- `DepthManager.createClassChildAtDepth()`
- `DepthManager.setDepthAbove()`
- `DepthManager.setDepthBelow()`
- `DepthManager.setDepthTo()`

Los métodos siguientes componen la interfaz API de espacio de profundidad reservada:

- `DepthManager.createClassObjectAtDepth()`
- `DepthManager.createObjectAtDepth()`

Resumen de métodos de la clase DepthManager

En la tabla siguiente se enumeran los métodos de la clase DepthManager.

Método	Descripción
`DepthManager.createChildAtDepth()`	Crea un elemento secundario del símbolo especificado en la profundidad especificada.
`DepthManager.createClassChildAtDepth()`	Crea un objeto de la clase especificada en la profundidad especificada.
`DepthManager.createClassObjectAtDepth()`	Crea una instancia de la clase especificada en la profundidad especificada en el clip especial de profundidad máxima.
`DepthManager.createObjectAtDepth()`	Crea un objeto en la profundidad especificada en el clip de profundidad máxima.
`DepthManager.setDepthAbove()`	Establece la profundidad por encima de la instancia especificada.
`DepthManager.setDepthBelow()`	Establece la profundidad por debajo de la instancia especificada.
`DepthManager.setDepthTo()`	Establece la profundidad en la instancia especificada en el clip de profundidad máxima.

Resumen de propiedades de la clase DepthManager

En la siguiente tabla se enumeran las propiedades de la clase DepthManager. Los valores constantes que se muestran son valores que el algoritmo DepthManager utiliza para organizar la profundidad. Si realiza un seguimiento de las siguientes propiedades, verá esos valores constantes en el panel Salida.

Sin embargo, cuando se ha implementado un método de DepthManager como `DepthManager.setDepthTo()`, utilizando una de las siguientes propiedades y, a continuación, se realiza un seguimiento de la profundidad del componente o del clip de película, DepthManager define las profundidades en incrementos de 20. El algoritmo incrementa las profundidades en caso de que Flash necesite insertar algo más en el medio, en función de otros scripts, componentes o demás.

Propiedad	Descripción
`DepthManager.kBottom`	Propiedad estática con el valor constante 202.
`DepthManager.kCursor`	Propiedad estática con el valor constante 101. Es la profundidad del cursor.
`DepthManager.kNotopmost`	Propiedad estática con el valor constante 204.

Propiedad	Descripción
DepthManager.kTooltip	Propiedad estática con el valor constante 102. Es la profundidad de la sugerencia.
DepthManager.kTop	Propiedad estática con el valor constante 201.
DepthManager.kTopmost	Propiedad estática con el valor constante 203.

DepthManager.createChildAtDepth()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

movieClipInstance.createChildAtDepth(*linkageName*, *depthFlag*[, *initObj*])

Parámetros

linkageName Identificador de vinculación. Este parámetro es una cadena.

depthFlag Uno de los valores siguientes: DepthManager.kTop, DepthManager.kBottom, DepthManager.kTopmost, DepthManager.kNotopmost. Todas las etiquetas de profundidad son propiedades estáticas de la clase DepthManager. Debe hacer referencia al paquete DepthManager (por ejemplo, mx.managers.DepthManager.kTopmost), o utilizar la sentencia import para importar el paquete DepthManager.

initObj Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al objeto creado. El tipo devuelto es MovieClip.

Descripción

Método; crea una instancia secundaria del símbolo que especifica *linkageName* en la profundidad que especifica *depthFlag*.

Ejemplo

El siguiente ejemplo crea una instancia minuteHand del clip de película MinuteSymbol y la coloca delante del reloj:

```
import mx.managers.DepthManager;
minuteHand = clock.createChildAtDepth("MinuteSymbol", DepthManager.kTop);
```

DepthManager.createClassChildAtDepth()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

movieClipInstance.createClassChildAtDepth(*className*, *depthFlag*[, *initObj*])

Parámetros

className Nombre de clase. Este parámetro es de tipo Function.

depthFlag Uno de los valores siguientes: DepthManager.kTop, DepthManager.kBottom, DepthManager.kTopmost, DepthManager.kNotopmost. Todas las etiquetas de profundidad son propiedades estáticas de la clase DepthManager. Debe hacer referencia al paquete DepthManager (por ejemplo, mx.managers.DepthManager.kTopmost), o utilizar la sentencia import para importar el paquete DepthManager.

initObj Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al elemento secundario creado. El tipo devuelto es UIObject.

Descripción

Método; crea una instancia secundaria de la clase que especifica *className* en la profundidad que especifica *depthFlag*.

Ejemplo

El siguiente código dibuja un rectángulo de selección delante de todos los objetos NoTopmost:

```
import mx.managers.DepthManager
this.ring = createClassChildAtDepth(mx.skins.RectBorder,
  DepthManager.kTop);
```

El siguiente código crea una instancia de la clase Button y le atribuye un valor a su propiedad label como parámetro *initObj*:

```
import mx.managers.DepthManager
button1 = createClassChildAtDepth(mx.controls.Button, DepthManager.kTop,
  {label: "Top Button"});
```

DepthManager.createClassObjectAtDepth()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.createClassObjectAtDepth(className, depthSpace[, initObj])
```

Parámetros

`className` Nombre de clase. Este parámetro es de tipo Function.

`depthSpace` Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase DepthManager. Debe hacer referencia al paquete DepthManager (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete DepthManager.

`initObj` Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al objeto creado. El tipo devuelto es UIObject.

Descripción

Método; crea un objeto de la clase que especifica *`className`* en la profundidad que especifica *`depthSpace`*. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

Ejemplo

En el ejemplo siguiente se crea un objeto de la clase Button:

```
import mx.managers.DepthManager
myCursorButton = Detph.createClassObjectAtDepth(mx.controls.Button,
  DepthManager.kCursor, {label: "Cursor"});
```

DepthManager.createObjectAtDepth()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.createObjectAtDepth(linkageName, depthSpace[, initObj])
```

Parámetros

linkageName Identificador de vinculación. Este parámetro es de tipo String.

depthSpace Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase DepthManager. Debe hacer referencia al paquete DepthManager (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete DepthManager.

initObj Objeto de inicialización opcional.

Valor devuelto

Referencia al objeto creado. El tipo devuelto es MovieClip.

Descripción

Método; crea un objeto en la profundidad especificada. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

Ejemplo

En el ejemplo siguiente se crea una instancia del símbolo TooltipSymbol y se coloca en la profundidad reservada para las sugerencias:

```
import mx.managers.DepthManager
myCursorTooltip = DepthManager.createObjectAtDepth("TooltipSymbol",
  DepthManager.kTooltip);
```

DepthManager.kBottom

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kBottom
```

Descripción

Propiedad (estática); propiedad con el valor constante 202. Esta propiedad se pasa como un parámetro en llamadas a `DepthManager.createClassChildAtDepth()` y `DepthManager.createChildAtDepth()` para colocar contenido detrás de otro contenido.

DepthManager.kCursor

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kCursor
```

Descripción

Propiedad (estática); propiedad con el valor constante 101. Esta propiedad se pasa como un parámetro en llamadas a `DepthManager.createClassObjectAtDepth()` y `DepthManager.createObjectAtDepth()` para solicitar la colocación en la profundidad del cursor.

DepthManager.kNotopmost

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kNotopmost
```

Descripción

Propiedad (estática); propiedad con el valor constante 204. Esta propiedad se pasa como un parámetro en llamadas a `DepthManager.createClassChildAtDepth()` y `DepthManager.createChildAtDepth()` para solicitar la eliminación de la capa superior.

DepthManager.kTooltip

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kTooltip
```

Descripción

Propiedad (estática); propiedad con el valor constante 102. Esta propiedad se pasa como un parámetro en llamadas a `DepthManager.createClassObjectAtDepth()` y `DepthManager.createObjectAtDepth()` para colocar un objeto en la profundidad de la sugerencia.

DepthManager.kTop

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kTop
```

Descripción

Propiedad (estática); propiedad con el valor constante 201. Esta propiedad se pasa como un parámetro en llamadas a `DepthManager.createClassChildAtDepth()` y `DepthManager.createChildAtDepth()` para solicitar la colocación encima de otro contenido pero debajo del contenido `DepthManager.kTopmost`.

DepthManager.kTopmost

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
DepthManager.kTopmost
```

Descripción

Propiedad (estática); propiedad con el valor constante 203. Esta propiedad se utiliza en llamadas a `DepthManager.createClassChildAtDepth()` y `DepthManager.createChildAtDepth()` para solicitar la colocación encima de otro contenido, incluidos los objetos `DepthManager.kTop`.

DepthManager.setDepthAbove()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
movieClipInstance.setDepthAbove(instance)
```

Parámetros

instance Nombre de instancia. Este parámetro es de tipo MovieClip.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por encima de la profundidad de la instancia que especifica el parámetro *instance* y mueve otros objetos si es necesario.

DepthManager.setDepthBelow()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
movieClipInstance.setDepthBelow(instance)
```

Parámetros

instance Nombre de instancia. Este parámetro es de tipo MovieClip.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por debajo de la profundidad de la instancia especificada y mueve otros objetos si es necesario.

Ejemplo

El código siguiente establece la profundidad de la instancia `textInput` por debajo de la profundidad de `button`:

```
textInput.setDepthBelow(button);
```

DepthManager.setDepthTo()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`movieClipInstance``.setDepthTo(`*`depthFlag`*`)`

Parámetros

`depthFlag` Uno de los valores siguientes: `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Todas las etiquetas de profundidad son propiedades estáticas de la clase DepthManager. Debe hacer referencia al paquete DepthManager (por ejemplo, `mx.managers.DepthManager.kTopmost`), o utilizar la sentencia `import` para importar el paquete DepthManager.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de *`movieClipInstance`* en el valor que especifica *`depthFlag`*. Este método desplaza una instancia a otra profundidad para dar cabida a otro objeto. DepthManager utiliza un algoritmo de ordenación aleatoria para establecer las profundidades en incrementos de 20. El algoritmo incrementa las profundidades en caso de que Flash necesite insertar algo más en el medio, en función de otros scripts, componentes o demás.

Ejemplo

En el siguiente ejemplo se utilizan dos componentes (o clips de película) para aumentar su profundidad alternativamente en incrementos de 20 cuando se hace clic en cada uno de ellos. Primero añada un componente Button al escenario y asígnele el nombre de instancia `a_btn`. A continuación, añada otro componente Button al escenario y asígnele el nombre de instancia `b_btn`. Asegúrese de que los botones se superponen de la siguiente manera:


```
import mx.managers.DepthManager;

a_btn.onRelease = function() {
b_btn.setDepthTo(DepthManager.kTop);
var b_depth:Number = b_btn.getDepth();
trace(b_depth);
}

b_btn.onRelease = function() {
a_btn.setDepthTo(DepthManager.kTop);
var a_depth:Number = a_btn.getDepth();
trace(a_depth);
}
```

Comprobación del archivo SWF. Cuando hace clic en el botón superior, el otro botón cambia de profundidad y se mueve hacia delante, y el panel Salida muestra la profundidad de ese botón. Los valores son 20, a continuación, 40, 60, etc.; se incrementan en 20 cada vez que se hace clic.

NOTA	Si utiliza DepthManager con instancias de clip de película en lugar de instancias de componente, es posible que necesite añadir un componente de IU a la biblioteca (si aún no hay ninguno) para que DepthManager funcione correctamente. DepthManager necesita un componente en el escenario o en la biblioteca para poder funcionar.

Para más información sobre la profundidad y el orden de apilamiento, consulte "Determinación del siguiente valor superior de profundidad disponible" en *Aprendizaje de ActionScript 2.0 en Flash*.

CAPÍTULO 21

Clase EventDispatcher

21

Los eventos permiten a la aplicación saber cuándo interactúa el usuario con un componente y cuándo se producen cambios importantes en el aspecto o ciclo de vida del componente, como por ejemplo su creación, destrucción o cambio de tamaño.

Los métodos de la clase EventDispatcher permiten añadir y eliminar detectores de eventos para que el código de usuario pueda reaccionar a los eventos de forma apropiada. Por ejemplo, se utiliza el método `EventDispatcher.addEventListener()` para registrar un detector con una instancia de componente. El detector se invoca cuando se activa un evento del componente.

Si se desea escribir un objeto personalizado que emita eventos que no están relacionados con la interfaz de usuario, es más rápido y corto utilizar EventDispatcher en combinación con UIComponent que con UIEventDispatcher.

Objetos de evento

Un objeto de evento pasa al detector como un parámetro. El objeto de evento es un objeto de ActionScript que tiene propiedades que contienen información acerca del evento que ha tenido lugar. El objeto de evento se puede utilizar dentro de la función callback del detector para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento. Por ejemplo, el código siguiente utiliza la propiedad `target` del objeto de evento `evtObj` para acceder a la propiedad `label` de la instancia `myButton` y enviar el valor al panel Salida:

```
listener = new Object();
listener.click = function(evtObj){
  trace("The " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Algunas propiedades de objeto de evento se definen en la especificación W3C (www.w3.org/TR/DOM-Level-3-Events/events.html) pero no se implementan en la versión 2 de la arquitectura de componentes de Macromedia. Cada objeto de evento de la versión 2 tiene las propiedades que se especifican en la tabla que se muestra a continuación. Algunos eventos tienen propiedades adicionales definidas y, en este caso, las propiedades se enumeran en la entrada del evento.

Propiedad	Descripción
`type`	Cadena que indica el nombre del evento.
`target`	Referencia a la instancia del componente que difunde el evento.

Clase EventDispatcher (API)

Nombre de clase de ActionScript mx.events.EventDispatcher

Resumen de métodos de la clase EventDispatcher

En la tabla siguiente se enumeran los métodos de la clase EventDispatcher.

Método	Descripción
`EventDispatcher.addEventListener()`	Registra un detector con una instancia de componente.
`EventDispatcher.dispatchEvent()`	Distribuye un evento mediante programación.
`EventDispatcher.removeEventListener()`	Elimina un detector de eventos de una instancia de componente.

EventDispatcher.addEventListener()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

componentInstance.addEventListener(*event*, *listener*)

Parámetros

event Cadena que es el nombre del evento.

listener Referencia a un objeto detector o a una función.

Valor devuelto

Ninguno.

Descripción

Método; registra un objeto detector con una instancia de componente que difunde un evento. Cuando se produce el evento, se notifica al objeto detector o a la función. Se puede invocar este método desde cualquier instancia de componente. Por ejemplo, el código siguiente registra un detector para la instancia de componente `myButton`:

```
myButton.addEventListener("click", myListener);
```

Es necesario definir el detector, tanto si es un objeto como una función, antes de llamar a `addEventListener()` para registrar el detector con la instancia de componente. Si el detector es un objeto, debe tener definida una función callback que se invoque al producirse el evento. Por lo general, la función callback tiene el mismo nombre que el evento con el que se ha registrado el detector. Si el detector es una función, ésta se invoca cuando se produce el evento. Para más información, consulte "Utilización de detectores para gestionar eventos" en *Utilización de componentes*.

Es posible registrar varios detectores en una sola instancia de componente, pero es preciso utilizar una llamada individual a `addEventListener()` para cada detector. Asimismo, se puede registrar un detector en varias instancias de componente, pero hay que utilizar una llamada individual a `addEventListener()` para cada instancia. Por ejemplo, el siguiente código define un objeto detector y lo asigna a dos instancias de componente Button cuyas propiedades `label` son `button1` y `button2`, respectivamente:

```
lo = new Object();
lo.click = function(evt){
  trace(evt.target.label + " clicked");
}
button1.addEventListener("click", lo);
button2.addEventListener("click", lo);
```

El orden de ejecución no está garantizado. No espere que se llame a un detector antes que a otro.

Un objeto de evento pasa al detector como un parámetro. El objeto de evento tiene propiedades que contienen información sobre el evento producido. Es posible utilizar el objeto de evento dentro de la función callback del detector para acceder a la información sobre el tipo de evento que tuvo lugar y la instancia que difundió el evento. En el ejemplo anterior, el objeto de evento es `evt` (se puede utilizar cualquier identificador como nombre del objeto de evento) y se utiliza en las sentencias `if` para determinar la instancia de botón sobre la que se ha hecho clic. Para más información, consulte "El objeto de evento" en *Utilización de componentes*.

Ejemplo

En el ejemplo siguiente se define un objeto detector, `myListener`, y la función callback para el evento `click`. A continuación, llama a `addEventListener()` para registrar el objeto detector `myListener` con la instancia de componente `myButton`.

```
myListener = new Object();
myListener.click = function(evt){
  trace(evt.type + " triggered");
}
myButton.addEventListener("click", myListener);
```

Para probar este código, coloque un componente Button en el escenario con el nombre de instancia `myButton` e introduzca este código en el fotograma 1.

EventDispatcher.dispatchEvent()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

`dispatchEvent(`*`eventObject`*`)`

Parámetros

`eventObject` Referencia a un objeto de evento. El objeto de evento debe tener una propiedad `type` que es una cadena que indica el nombre del evento. Normalmente, el objeto de evento también tiene una propiedad `target` que es el nombre de la instancia que difunde el evento. Se pueden definir otras propiedades en el objeto de evento que ayudarán al usuario a capturar información acerca del evento cuando éste se distribuye.

Valor devuelto

Ninguno.

Descripción

Método; distribuye un evento a cualquier detector registrado con una instancia de la clase. Este método suele llamarse desde un archivo de clase del componente. Por ejemplo, el archivo de clase SimpleButton.as distribuye el evento `click`.

Ejemplo

En el siguiente ejemplo se distribuye un evento `click`:

```
dispatchEvent({type:"click"});
```

EventDispatcher.removeEventListener()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

componentInstance.removeEventListener(*event*, *listener*)

Parámetros

event Cadena que es el nombre del evento.

listener Referencia a un objeto detector o a una función.

Valor devuelto

Ninguno.

Descripción

Método; quita el registro de un objeto detector desde una instancia de componente que difunde un evento.

CAPÍTULO 22

Componente FLVPlayback (sólo en Flash Professional)

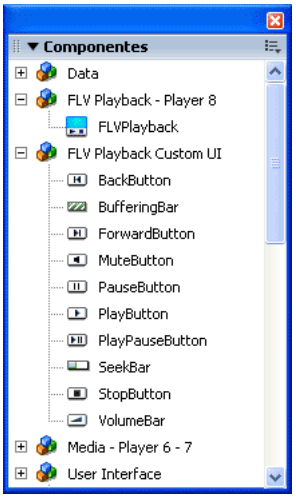
22

El componente FLVPlayback permite incluir fácilmente un reproductor de vídeo en la aplicación Flash para reproducir archivos de vídeo Flash (FLV) descargados de forma progresiva a través de HTTP, o reproducir archivos FLV sin interrupción, desde Flash Communication Server (FCS) o desde Flash Video Streaming Service (FVSS).

El componente FLVPlayback, de uso sencillo, tiene las siguientes características y ventajas:

- Puede arrastrarse al escenario e implementarse rápidamente de forma correcta
- Proporciona una colección de *aspectos* prediseñados que permiten personalizar la apariencia de los controles de reproducción
- Permite a los usuarios avanzados crear sus propios aspectos
- Proporciona cuepoints que permiten sincronizar el vídeo con el texto, los gráficos y la animación
- Proporciona una previsualización dinámica de las personalizaciones
- Mantiene un tamaño de archivo SWF razonable

El componente FLVPlayback incluye los componentes de interfaz de usuario personalizados de reproducción FLV. El componente FLVPlayback es una combinación del área de visualización, o reproductor de vídeo, donde se visualiza el archivo FLV y los controles que permiten que funcione. Los componentes de interfaz de usuario personalizados de reproducción FLV proporcionan botones de control y mecanismos que puede utilizar para reproducir, detener, pausar o controlar de alguna otra forma el archivo FLV. Estos controles son: BackButton, BufferingBar, ForwardButton, MuteButton, PauseButton, PlayButton, PlayPauseButton, SeekBar, StopButton y VolumeBar. El componente FLVPlayback y los controles personalizados de interfaz de usuario de reproducción FLV aparecen en el panel Componentes, como se indica en la siguiente ilustración:



El proceso de añadir controles de reproducción al componente FLVPlayback se denomina *aplicación de aspectos*. El componente FLVPlayback tiene un aspecto predeterminado inicial, ClearOverPlaySeekMute.swf, que proporciona controles transparentes para las funciones de reproducción, búsqueda y silencio. Para cambiar este aspecto, puede optar entre:

- Seleccionarlo desde una colección de aspectos prediseñados
- Seleccionar controles individuales desde los componentes de interfaz de usuario personalizados de reproducción FLV y personalizarlos
- Crear un aspecto personalizado y añadirlo a la colección de aspectos prediseñados

Cuando se selecciona un aspecto distinto, el aspecto seleccionado se convierte en el nuevo aspecto predeterminado.

Para más información sobre cómo seleccionar o crear un aspecto para el componente FLVPlayback, consulte "Personalización del componente FLVPlayback" en la página 541.

El componente FLVPlayback también incluye una interfaz de programación de aplicaciones (API) ActionScript. Esta API incluye las clases FLVPlayback, VideoError y VideoPlayer. Para más información sobre estas clases, consulte "Clase FLVPlayback" en la página 557, "Clase VideoPlayer" en la página 729 y "Clase VideoError" en la página 722.

Utilización del componente FLVPlayback

El uso del componente FLVPlayback consiste básicamente en colocarlo en el escenario y especificar un archivo FLV para que lo reproduzca. Sin embargo, también pueden establecerse diversos parámetros que especifiquen su comportamiento y describan el archivo FLV.

Creación de aplicaciones con el componente FLVPlayback

Puede incluir el componente FLVPlayback en la aplicación de las maneras siguientes:

- Arrastre el componente FLVPlayback desde el panel Componentes al escenario y especifique un valor para el parámetro `contentPath`.
- Use el Asistente de importación de vídeo para crear el componente en el escenario y personalizarlo eligiendo un aspecto.
- Utilice el método `attachMovie()` de MovieClip para crear dinámicamente una instancia de FLVPlayback en el escenario, suponiendo que el componente se encuentre en la biblioteca.

Para arrastrar el componente FLVPlayback desde el panel Componentes:

1. En el panel Componentes, haga clic en el signo más (+) para abrir la entrada FLV Playback - Player 8.
2. Arrastre el componente FLVPlayback al escenario.
3. Con el componente FLVPlayback seleccionado en el escenario, busque la celda Valor del parámetro `contentPath` en la ficha Parámetros del inspector de componentes e introduzca una cadena que especifique una de las opciones siguientes:
   - Una ruta de acceso local a un archivo FLV
   - Una URL de un archivo FLV
   - Una URL de un archivo XML que describa la forma de reproducir un archivo FLV

   Para más información sobre cómo crear un archivo XML para describir uno o varios archivos FLV, consulte "Utilización de un archivo SMIL" en la página 735.
4. En la ficha Parámetros del inspector de componentes, con el componente FLVPlayback seleccionado en el escenario, haga clic en la celda Valor para el parámetro `skin`.
5. Haga clic en el icono de lupa para abrir el cuadro de diálogo Seleccionar aspecto.

6. Seleccione una de las opciones siguientes:
   - En la lista desplegable Aspecto, seleccione uno de los aspectos prediseñados para asociar un conjunto de controles de reproducción al componente.
   - Si ha creado un aspecto personalizado, seleccione URL de aspecto personalizado en el menú desplegable e introduzca en el cuadro de texto URL la URL del archivo SWF que contiene el aspecto.
   - Seleccione Ninguno y arrastre componentes individuales de interfaz de usuario personalizados de reproducción FLV al escenario para añadir controles de reproducción.

   NOTA En los primeros dos casos, aparece una previsualización del aspecto en el panel de visualización situado encima del menú emergente.

7. Haga clic en Aceptar para cerrar el cuadro de diálogo Seleccionar aspecto.
8. Seleccione Probar película en el menú Control para ejecutar el archivo SWF e iniciar el vídeo.

Para utilizar el Asistente de importación de vídeo:

1. Seleccione Archivo > Importar > Importar vídeo.
2. Seleccione una de las siguientes opciones para indicar la ubicación del archivo de vídeo:
   - En mi equipo local
   - Ya se ha implementado en un servidor Web, Flash Video Streaming Service o Flash Communication Server
3. En función de su elección, deberá especificar la ruta del archivo o la URL que especifica la ubicación del archivo de vídeo. A continuación, haga clic en Siguiente.
4. Si ha seleccionado una ruta de archivo, aparecerá un cuadro de diálogo Implementación, donde puede seleccionar una de las opciones enumeradas para especificar cómo desea implementar el vídeo:
   - Descarga progresiva desde un servidor Web estándar
   - Flujo de Flash Video Streaming Service
   - Flujo de Flash Communication Server
   - Incorporar vídeo en SWF y reproducir en la línea de tiempo

   ADVERTENCIA No elija la opción de incorporar vídeo. El componente FLVPlayback sólo reproduce flujo de vídeo externo. Esta opción no colocará un componente FLVPlayback en el escenario.

5. Haga clic en Siguiente.
6. Seleccione una de las opciones siguientes:
   - En la lista desplegable Aspecto, seleccione uno de los aspectos prediseñados para asociar un conjunto de controles de reproducción al componente.
   - Si ha creado un aspecto personalizado para el componente, seleccione URL de aspecto personalizado en el menú emergente e introduzca en el cuadro de texto URL la URL del archivo SWF que contiene el aspecto.
   - Seleccione Ninguno y arrastre componentes individuales de interfaz de usuario personalizados de reproducción FLV al escenario para añadir controles de reproducción.

   NOTA En los primeros dos casos, aparece una previsualización del aspecto en el panel de visualización situado encima del menú emergente.
7. Haga clic en Aceptar para cerrar el cuadro de diálogo Seleccionar aspecto.
8. Lea el mensaje del cuadro de diálogo Finalizar importación de vídeo para ver qué sucederá a continuación y después haga clic en Finalizar.
9. Si no ha guardado el archivo FLA, aparecerá un cuadro de diálogo Guardar como.
10. Seleccione Probar película en el menú Control para ejecutar el archivo SWF e iniciar el vídeo.

Para crear dinámicamente una instancia mediante código ActionScript:

1. Arrastre el componente FLVPlayback desde el panel Componentes a la Biblioteca (Ventana > Biblioteca).
2. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. Cambie *install_drive* por la unidad en la que instaló Flash 8 y modifique la ruta de acceso para reflejar la ubicación de la carpeta Skins en la instalación:

```
import mx.video.*;
this.attachMovie("FLVPlayback", "my_FLVPlybk", 10, {width:320,
  height:240, x:100, y:100});
my_FLVPlybk.skin = "file:///install_drive|/Program Files/Macromedia/
  Flash 8/en/Configuration/Skins/ClearOverPlaySeekMute.swf"
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

   El método `attachMovie()` pertenece a la clase MovieClip. Puede utilizarlo para crear una instancia del componente FLVPlayback, ya que la clase FLVPlayback amplía la clase MovieClip.

   NOTA Si no se establecen las propiedades `contentPath` y `skin`, el clip de película generado aparecerá vacío.
3. Seleccione Probar película en el menú Control para ejecutar el archivo SWF e iniciar el archivo FLV.

Parámetros del componente FLVPlayback

Para cada instancia del componente FLVPlayback puede establecer los siguientes parámetros en el inspector de propiedades o en el inspector de componentes:

autoPlay Valor booleano que determina cómo se debe reproducir el archivo FLV. Si es `true`, el componente reproduce el archivo FLV inmediatamente después de cargarlo. Si es `false`, el componente carga el primer fotograma y pausa la reproducción. El valor predeterminado es `true` en el reproductor de vídeo predeterminado (0) y `false` en los demás reproductores. Para más información sobre el uso de varios reproductores de vídeo en una sola instancia de FLVPlayback, consulte "Reproducción de varios archivos FLV" en la página 538.

autoRewind Valor booleano que determina si el archivo FLV se rebobinará automáticamente cuando finalice su reproducción. Si es `true`, el componente FLVPlayback rebobina automáticamente el archivo FLV hasta el principio cuando la cabeza lectora llega al final o cuando el usuario hace clic en el botón Detener. Si es `false`, el componente detiene la reproducción en el último fotograma del archivo FLV y no lo rebobina automáticamente. El valor predeterminado es `true`.

autoSize Valor booleano. Si es `true`, cambia el tamaño del componente en tiempo de ejecución para usar las dimensiones del archivo FLV de origen. Estas dimensiones se codifican en el archivo FLV y difieren de las dimensiones predeterminadas del componente FLVPlayback. El valor predeterminado es `false`. Para más información, consulte `FLVPlayback.autoSize` en la página 577.

bufferTime Número de segundos que el archivo FLV se almacenará en búfer antes de que se inicie la reproducción. Este parámetro afecta a los archivos FLV sin interrupción, que se almacenan en búfer pero no se descargan. En los archivos FLV que se descargan progresivamente a través de HTTP, aumentar este valor no supone una gran ventaja, aunque sí puede mejorar la visualización de vídeo de alta calidad en un equipo antiguo que sea más lento. El valor predeterminado es 0,1. Para más información, consulte `FLVPlayback.bufferTime` en la página 589.

NOTA	Establecer este parámetro no garantiza que una parte del archivo FLV se descargue antes de iniciar la reproducción.

contentPath Cadena que especifica la URL de un archivo FLV o de un archivo XML que describe la forma de reproducir uno o varios archivos FLV. Puede especificar una ruta en el equipo local, una ruta HTTP o una ruta de protocolo de mensajería en tiempo real (RTMP). Haga doble clic en la celda Valor para que este parámetro abra el cuadro de diálogo Ruta del contenido. El valor predeterminado es una cadena vacía.

Si no especifica un valor para el parámetro `contentPath` no sucederá nada cuando Flash ejecute la instancia de FLVPlayback. Para más información, consulte "Especificación del parámetro contentPath" en la página 528.

cuePoints Cadena que describe los cuepoints para el archivo FLV. Los cuepoints permiten sincronizar puntos específicos del archivo FLV con animaciones, gráficos o texto de Flash. El valor predeterminado es una cadena vacía. Para más información, consulte "Utilización de cuepoints" en la página 530.

isLive Valor booleano. Si es `true`, especifica que el archivo FLV se está transmitiendo de forma dinámica desde Flash Communication Server. Un ejemplo de flujo en vivo es un vídeo de eventos de noticias que se transmiten mientras suceden. El valor predeterminado es `false`. Para más información, consulte `FLVPlayback.isLive` en la página 621.

maintainAspectRatio Valor booleano. Si es `true`, cambia el tamaño del reproductor de vídeo en el componente FLVPlayback para conservar la proporción del archivo FLV de origen; el archivo FLV de origen cambia su tamaño para ajustarse a las dimensiones del componente FLVPlayback en el escenario. El parámetro `autoSize` tiene prioridad sobre este parámetro. El valor predeterminado es `true`. Para más información, consulte `FLVPlayback.maintainAspectRatio` en la página 625.

skin Parámetro que abre el cuadro de diálogo Seleccionar aspecto, donde es posible elegir un aspecto para el componente. El valor predeterminado es inicialmente un aspecto prediseñado, pero posteriormente se convierte en el último aspecto seleccionado. Si selecciona `None`, la instancia de FLVPlayback no tiene elementos de control del archivo FLV. Si se establece el valor del parámetro `autoPlay` en `true`, el archivo FLV se reproduce automáticamente. Para más información, consulte "Personalización del componente FLVPlayback" en la página 541.

skinAutoHide Valor booleano. Si es `true`, oculta el aspecto cuando el puntero del ratón no se encuentra sobre el archivo FLV o la región de aspecto, si se trata de un aspecto externo que no se encuentra en el área del visualización del archivo FLV. El valor predeterminado es `false`. Para más información, consulte `FLVPlayback.skin` en la página 694.

totalTime Número total de segundos, con una precisión de milisegundos, en el archivo FLV de origen. El valor predeterminado es 0.

Si utiliza FCS o FVSS, el componente siempre toma el valor del tiempo total del servidor.

Si se utiliza la descarga progresiva a través de HTTP, el componente utiliza este valor en caso de que sea mayor que cero. De lo contrario, intenta utilizar el tiempo de los metadatos del archivo FLV. Para más información, consulte `FLVPlayback.totalTime` en la página 707.

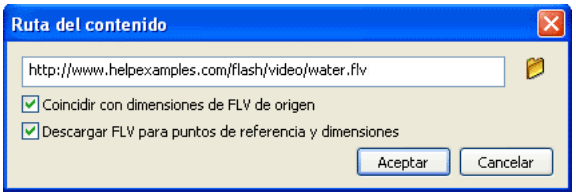
volume Número de 0 a 100 que representa el porcentaje del volumen máximo (100). Para más información, consulte `FLVPlayback.volume` en la página 713.

Cada uno de estos parámetros tiene una propiedad equivalente en la clase FLVPlayback. El valor establecido en esta propiedad sustituye el valor del parámetro en el inspector de componentes o el inspector de propiedades.

Especificación del parámetro contentPath

El parámetro `contentPath` permite especificar el nombre y la ubicación del archivo FLV, que informan a Flash sobre el modo de reproducir el archivo.

Abra el cuadro de diálogo Ruta del contenido haciendo doble clic en la celda Valor del parámetro `contentPath` del inspector de componentes. El cuadro de diálogo tiene el siguiente aspecto:



El cuadro de diálogo contiene dos casillas de verificación que permiten determinar las dimensiones de la instancia de FLVPlayback y especificar si deben obtenerse las dimensiones y la información de cuepoint del archivo FLV. Para más información, consulte "Opciones de archivos FLV" en la página 529.

Ruta del contenido

Introduzca la URL o la ruta de acceso local del archivo FLV o de un archivo XML que describa la forma de reproducir el archivo FLV. Si no conoce la ubicación exacta del archivo FLV, haga clic en el icono de carpeta para abrir un cuadro de diálogo Navegador que le permitirá buscar la ubicación correcta. Al buscar un archivo FLV, si está en la ubicación del archivo SWF de destino (o en una subcarpeta), Flash utiliza automáticamente la ruta de acceso relativa a esa ubicación, por lo que puede utilizarlo desde un servidor Web. De lo contrario será una ruta completa de Windows o Macintosh. Para introducir el nombre de un archivo XML local, debe escribir la ruta de acceso y el nombre.

Si especifica una URL HTTP, el archivo FLV se reproduce como un archivo de descarga progresiva. Si especifica una URL que sea RTMP, el archivo FLV se transmite desde un servidor FCS o FVSS. Una URL a un archivo XML también podría ser un archivo FLV transmitido sin interrupción desde un servidor FCS o FVSS.

ATENCIÓN	Cuando se hace clic en Aceptar en el cuadro de diálogo Ruta del contenido, el componente actualiza el valor del parámetro `cuePoints` porque es posible que ya no se aplique si cambia la ruta del contenido. En consecuencia, *podría* perder los cuepoints desactivados (aunque se conservarán los objetos cuepoint de ActionScript). No perderá los cuepoints desactivados si el nuevo archivo FLV contiene los mismos cuepoints, lo cual puede ocurrir si simplemente cambia la ruta de acceso. Por esta razón, es posible que desee desactivar mediante ActionScript los objetos cuepoint que no sean de ActionScript, en lugar de hacerlo a través del cuadro de diálogo Cuepoints.

También puede especificar la ubicación de un archivo XML que describa la forma de reproducir varios flujos de archivos FLV para múltiples anchos de banda. El archivo XML utiliza el lenguaje SMIL (Syncrhonized Multimedia Integration Language) para describir los archivos FLV. Para ver una descripción del archivo SMIL XML, consulte "Utilización de un archivo SMIL" en la página 735.

También puede especificar el nombre y la ubicación del archivo FLV mediante la propiedad `FLVPlayback.contentPath` y los métodos `FLVPlayback.play()` y `FLVPlayback.load()` de ActionScript. Estas tres alternativas tienen prioridad sobre el parámetro `contentPath` en el inspector de componentes. Para más información, consulte `FLVPlayback.contentPath` en la página 597, `FLVPlayback.play()` en la página 641 y `FLVPlayback.load()` en la página 623.

Opciones de archivos FLV

El cuadro de diálogo Ruta del contenido contiene además dos opciones. La primera opción, Coincidir con dimensiones de FLV de origen, especifica si la instancia de FLVPlayback en el escenario debería coincidir con las dimensiones del archivo FLV de origen. El archivo FLV de origen contiene dimensiones de altura y anchura preferidas para la reproducción. Si selecciona la primera opción, las dimensiones de la instancia de FLVPlayback cambian de tamaño para que coincidan con las dimensiones preferidas. Sin embargo, esta opción sólo está disponible si también se selecciona la segunda opción.

La segunda opción, Descargar FLV para cuepoints y dimensiones, sólo puede activarse si la ruta del contenido es una URL HTTP o RTMP, lo que significa que el archivo FLV no es local. Cualquier ruta que no termine con .flv se considerará también una ruta de red porque debe ser un archivo XML y podría señalar a archivos FLV en cualquier ubicación. Esta opción especifica si debe descargarse o transmitirse una parte del archivo FLV para obtener las dimensiones del archivo FLV y cualquier definición de cuepoint que tenga incorporada. Flash utiliza las dimensiones para cambiar el tamaño de la instancia de FLVPlayback y carga las definiciones de cuepoint en el parámetro `cuePoints` del inspector de componentes. Si no se activa esta opción, la primera opción aparece desactivada.

Utilización de cuepoints

Un cuepoint es un punto en el que el reproductor de vídeo distribuye un evento `cuePoint` mientras se reproduce un archivo FLV. Puede añadir cuepoints a un archivo FLV cuando desee interactuar con otro elemento de la página Web. Quizá desee mostrar texto o un gráfico, por ejemplo, o sincronizar con una animación de Flash, o pausar la reproducción del archivo FLV, buscar otro instante específico de la reproducción o cambiar a otro archivo FLV. Los cuepoints permiten recibir el control en el código ActionScript y sincronizar dichos puntos del archivo FLV con otras acciones de la página Web.

Hay tres tipos de cuepoints: de navegación, de evento y de ActionScript. Los cuepoints de navegación y eventos se denominan también cuepoints *incorporados* porque se incorporan en el flujo de archivos FLV y en el paquete de metadatos del archivo FLV.

Un *cuepoint de navegación* permite buscar un determinado fotograma en el archivo FLV, ya que crea un *fotograma clave* en el archivo FLV, lo más cerca posible al tiempo especificado. Un fotograma clave es un segmento de datos que se produce entre los fotogramas de imagen del flujo del archivo FLV. Cuando se busca un cuepoint de navegación, el componente busca el fotograma clave e inicia el evento `cuePoint`.

Un *cuepoint de evento* permite sincronizar un instante específico del archivo FLV con un evento externo de la página Web. El evento `cuePoint` se produce precisamente en el instante especificado. Puede incorporar cuepoints de navegación y de evento en un archivo FLV mediante el Asistente de importación de vídeo o Flash Video Encoder. Para más información sobre el Asistente de importación de vídeo y Flash Video Encoder, consulte Capítulo 11, “Trabajo con vídeo” en *Utilización de Flash*.

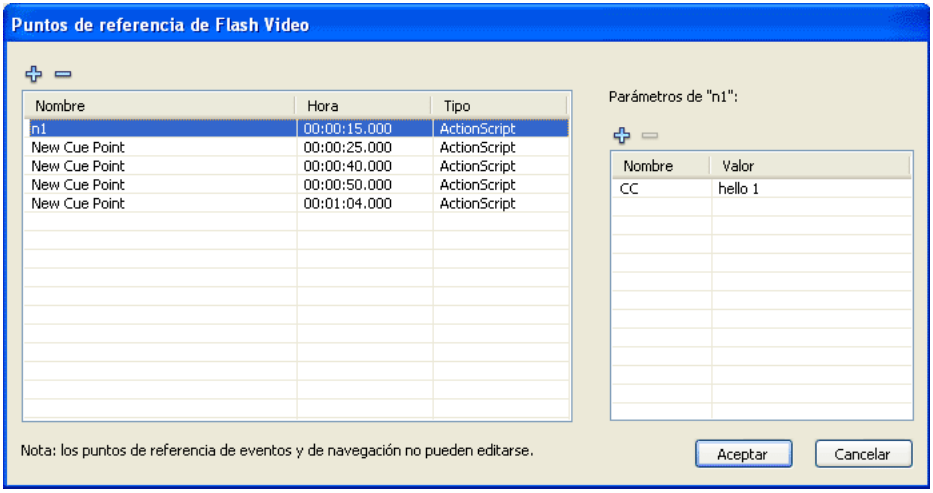
Un *cuepoint de ActionScript* es un cuepoint externo que puede añadirse a través del cuadro de diálogo Cuepoints de Flash Video del componente o a través del método `FLVPlayback.addASCuePoint()`. El componente almacena y rastrea los cuepoints de ActionScript independientemente del archivo FLV, por lo que son menos precisos que los cuepoints incorporados. La precisión de los cuepoints de ActionScript es de una décima de segundo. Para aumentar la precisión de los cuepoints de ActionScript, disminuya el valor de la propiedad `playheadUpdateInterval` porque el componente genera el evento `cuePoint` para los cuepoints de ActionScript cuando se actualiza la cabeza lectora. Para más información, consulte "FLVPlayback.playheadUpdateInterval" en la página 647.

En ActionScript y en los metadatos del archivo FLV, un cuepoint se representa como un objeto con las siguientes propiedades: `name`, `time`, `type` y `parameters`. La propiedad `name` es una cadena que contiene el nombre asignado del cuepoint. La propiedad `time` es un número que representa el tiempo en horas, minutos, segundos y milisegundos (HH:MM:SS.mmm) cuando se produce el cuepoint. La propiedad `type` es una cadena cuyo valor es `"navigation"`, `"event"` o `"actionscript"`, en función del tipo de cuepoint que haya creado. La propiedad `parameters` es una matriz de los pares nombre-valor especificados.

Cuando se produce un evento `cuePoint`, el objeto cuepoint está disponible en el objeto de evento a través de la propiedad `info`. Para más información, consulte "Detección de eventos cuePoint" en la página 534.

Utilización del cuadro de diálogo Puntos de referencia de Flash Video

Abra el cuadro de diálogo Puntos de referencia de Flash Video haciendo doble clic en la celda Valor del parámetro `cuePoints` del inspector de componentes. El cuadro de diálogo tiene el siguiente aspecto:



El cuadro de diálogo muestra los cuepoints de ActionScript e incorporados. Puede utilizar este cuadro de diálogo para añadir y eliminar cuepoints de ActionScript y parámetros de cuepoint. También puede activar o desactivar cuepoints incorporados. Sin embargo, no puede añadir, cambiar ni eliminar cuepoints incorporados.

Para añadir un cuepoint de ActionScript:

1. Haga doble clic en la celda Valor del parámetro `cuePoints` en el inspector de componentes para abrir el cuadro de diálogo Puntos de referencia de Flash Video.
2. Haga clic en el signo más (+) situado en la esquina superior izquierda, por encima de la lista de cuepoints, para añadir una entrada de cuepoint de ActionScript predeterminada.
3. Haga clic en el texto Nuevo cuepoint de la columna Nombre y edite el texto para asignar un nombre al cuepoint.
4. Haga clic en el valor de tiempo de 00:00:00:000 para editarlo y asigne un tiempo para el cuepoint. Puede especificar el tiempo en horas, minutos, segundos y milisegundos (HH:MM:SS.mmm).

   Si hay varios cuepoints, el cuadro de diálogo mueve el nuevo cuepoint a su posición cronológica en la lista.

5. Para añadir un parámetro para el cuepoint seleccionado, haga clic en el signo más (+) situado encima de la sección Parámetros e introduzca valores en las columnas Nombre y Valor. Repita este paso para cada parámetro.
6. Para añadir más cuepoints de ActionScript, repita los pasos 2 a 5 para cada uno.
7. Haga clic en Aceptar para guardar los cambios.

Para eliminar un cuepoint de ActionScript:

1. Haga doble clic en la celda Valor del parámetro `cuePoints` en el inspector de componentes para abrir el cuadro de diálogo Puntos de referencia de Flash Video.
2. Seleccione los cuepoints que desea eliminar.
3. Haga clic en el signo menos (-) situado en la esquina superior izquierda, por encima de la lista de cuepoints, para eliminarlo.
4. Repita los pasos 2 y 3 para cada cuepoint que desee eliminar.
5. Haga clic en Aceptar para guardar los cambios.

Para activar o desactivar un cuepoint de archivo FLV incorporado:

1. Haga doble clic en la celda Valor del parámetro `cuePoints` en el inspector de componentes para abrir el cuadro de diálogo Puntos de referencia de Flash Video.
2. Seleccione el cuepoint que desea activar o desactivar.
3. Haga clic en el valor de la columna Tipo para activar el menú emergente o haga clic en la flecha abajo.
4. Haga clic en el nombre del tipo de cuepoint (por ejemplo, Event o Navigation) para activarlo. Haga clic en Disabled para desactivarlo.
5. Haga clic en Aceptar para guardar los cambios.

Utilización de ActionScript con cuepoints

Puede utilizar ActionScript para añadir cuepoints de ActionScript, detectar eventos cuePoint, buscar cuepoints de cualquier tipo o de un tipo especificado, buscar un cuepoint de navegación, activar o desactivar un cuepoint, comprobar si un cuepoint está activado y quitar un cuepoint.

En los ejemplos de esta sección se utiliza un archivo FLV denominado cuepoints.flv, que contiene los tres cuepoints siguientes:

Nombre	Tiempo	Tipo
point1	00:00:00.418	Navigation
point2	00:00:07.748	Navigation
point3	00:00:16.020	Navigation

Adición de cuepoints de ActionScript

Puede añadir cuepoints de ActionScript a un archivo FLV mediante el método `addASCuePoint()`. En el siguiente ejemplo se añaden dos cuepoints de ActionScript al archivo FLV cuando está listo para reproducirse. Se añade el primer cuepoint mediante un objeto cuepoint, que especifica el tiempo, el nombre y el tipo del cuepoint en sus propiedades. En la segunda llamada se especifica el tiempo y el nombre mediante los parámetros `time` y `name` del método.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var cuePt:Object = new Object(); //crear objeto cuepoint
cuePt.time = 2.02;
cuePt.name = "ASpt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
// añadir segundo objeto cuepoint de AS mediante los parámetros time y name
my_FLVPlybk.addASCuePoint(5, "ASpt2");
```

Para más información, consulte `FLVPlayback.addASCuePoint()` en la página 570.

Detección de eventos cuePoint

El evento `cuePoint` permite recibir el control en el código ActionScript cuando se produce un evento `cuePoint`. Cuando se producen cuepoints en el siguiente ejemplo, el detector de `cuePoint` llama a una función de controlador de eventos que muestra el valor de la propiedad `playheadTime` y el nombre y el tipo del cuepoint.

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Elapsed time in seconds: " + my_FLVPlybk.playheadTime);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Para más información sobre el evento cuePoint, consulte `FLVPlayback.cuePoint` en la página 599.

Búsqueda de cuepoints

El código ActionScript permite encontrar un cuepoint de cualquier tipo, el cuepoint más cercano en el tiempo o el siguiente cuepoint con un nombre específico.

El controlador de eventos `ready` del siguiente ejemplo llama al método `findCuePoint()` para buscar el cuepoint `ASpt1` y, a continuación, llama al método `findNearestCuePoint()` para buscar el cuepoint de navegación más cercano en el tiempo al cuepoint `ASpt1`:

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var rtn_obj:Object = new Object(); //crear objeto cuepoint
my_FLVPlybk.addASCuePoint(2.02, "ASpt1");  //añadir objeto cuepoint de AS
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  rtn_obj = my_FLVPlybk.findCuePoint("ASpt1");
  traceit(rtn_obj);
  rtn_obj = my_FLVPlybk.findNearestCuePoint(rtn_obj.time,
  FLVPlayback.NAVIGATION);
  traceit(rtn_obj);
}
my_FLVPlybk.addEventListener("ready", listenerObject);
function traceit(cuePoint:Object):Void {
  trace("Cue point name is: " + cuePoint.name);
  trace("Cue point time is: " + cuePoint.time);
  trace("Cue point type is: " + cuePoint.type);
}
```

En el siguiente ejemplo, el controlador de eventos `ready` busca el cuepoint `ASpt` y llama al método `findNextCuePointWithName()` para buscar el siguiente cuepoint con el mismo nombre:

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var rtn_obj:Object = new Object(); //crear objeto cuepoint
my_FLVPlybk.addASCuePoint(2.02, "ASpt");  //añadir objeto cuepoint de AS
my_FLVPlybk.addASCuePoint(3.4, "ASpt");  //añadir segundo Aspt
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  rtn_obj = my_FLVPlybk.findCuePoint("ASpt");
  traceit(rtn_obj);
  rtn_obj = my_FLVPlybk.findNextCuePointWithName(rtn_obj);
  traceit(rtn_obj);
}
my_FLVPlybk.addEventListener("ready", listenerObject);
function traceit(cuePoint:Object):Void {
  trace("Cue point name is: " + cuePoint.name);
  trace("Cue point time is: " + cuePoint.time);
  trace("Cue point type is: " + cuePoint.type);
}
```

Para más información sobre cómo buscar cuepoints, consulte `FLVPlayback.findCuePoint()` en la página 605, `FLVPlayback.findNearestCuePoint()` en la página 608 y `FLVPlayback.findNextCuePointWithName()` en la página 611.

Búsqueda de cuepoints de navegación

Puede buscar un cuepoint de navegación y buscar el anterior y siguiente cuepoint de navegación a partir de un tiempo especificado. En el siguiente ejemplo se reproduce el archivo FLV cuepoints.flv y se busca el cuepoint correspondiente a 7.748 cuando se produce el evento `ready`. Cuando se produce el evento cuePoint, el ejemplo llama al método `seekToPrevNavCuePoint()` para buscar el primer cuepoint. Cuando se produce dicho evento `cuePoint`, el ejemplo llama al método `seekToNextNavCuePoint()` para buscar el último cuepoint añadiendo 10 segundos a `eventObject.info.time`, que es el tiempo del cuepoint actual.

```
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekToNavCuePoint("point2");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
  trace(eventObject.info.time);
  if(eventObject.info.time == 7.748)
    my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time - .005);
  else
    my_FLVPlybk.seekToNextNavCuePoint(eventObject.info.time + 10);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://helpexamples.com/flash/video/
  cuepoints.flv";
```

Para más información, consulte `FLVPlayback.seekToNavCuePoint()` en la página 683, `FLVPlayback.seekToNextNavCuePoint()` en la página 685 y `FLVPlayback.seekToPrevNavCuePoint()` en la página 687.

Activación y desactivación de cuepoints de archivo FLV incorporados

Puede activar y desactivar cuepoints de archivo FLV incorporados, mediante el método `setFLVCuePointEnabled()`. Los cuepoints desactivados no activan eventos `cuePoint` ni funcionan con los métodos `seekToCuePoint()`, `seekToNextNavCuePoint()` y `seekToPrevNavCuePoint()`. Sin embargo, puede buscar los cuepoints desactivados mediante los métodos `findCuePoint()`, `findNearestCuePoint()` y `findNextCuePointWithName()`.

Puede probar si un cuepoint de archivo FLV incorporado está activado, mediante el método `isFLVCuePointEnabled()`. En el siguiente ejemplo, se desactivan los cuepoints incorporados `point2` y `point3` cuando el vídeo está listo para reproducirse. Sin embargo, cuando se produce el primer evento `cuePoint`, el controlador de eventos prueba si el cuepoint `point3` está desactivado y, si lo está, lo activa.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
  my_FLVPlybk.setFLVCuePointEnabled(false, "point3");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point time is: " + eventObject.info.time);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
    if (my_FLVPlybk.isFLVCuePointEnabled("point2") == false) {
      my_FLVPlybk.setFLVCuePointEnabled(true, "point2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Para más información, consulte `FLVPlayback.isFLVCuePointEnabled()` en la página 619 y `FLVPlayback.setFLVCuePointEnabled()` en la página 690.

Eliminación de un cuepoint de ActionScript

Puede quitar un cuepoint de ActionScript mediante el método `removeASCuePoint()`. En el siguiente ejemplo, se quita el cuepoint `ASpt2` cuando se produce el cuepoint `ASpt1`:

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point name is: " + eventObject.info.name);
    if (eventObject.info.name == "ASpt1") {
      my_FLVPlybk.removeASCuePoint("ASpt2");
      trace("Removed cue point ASpt2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Para más información, consulte `FLVPlayback.removeASCuePoint()` en la página 658.

Reproducción de varios archivos FLV

Puede reproducir una secuencia de archivos FLV en una instancia de FLVPlayback; para ello, simplemente debe cargar una nueva URL en la propiedad `contentPath` cuando finalice la reproducción del anterior archivo FLV. Por ejemplo, el siguiente código ActionScript detecta el evento `complete`, que se produce cuando finaliza la reproducción de un archivo FLV. Cuando se produce este evento, el código establece el nombre y la ubicación del nuevo archivo FLV en la propiedad `contentPath` y llama al método `play()` para reproducir el nuevo vídeo.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
// detectar evento complete; reproducir nuevo archivo FLV
listenerObject.complete = function(eventObject:Object):Void {
  if (my_FLVPlybk.contentPath == "http://www.helpexamples.com/flash/video/
  clouds.flv") {
    my_FLVPlybk.play("http://www.helpexamples.com/flash/video/water.flv");
  }
};
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Utilización de varios reproductores de vídeo

También es posible abrir varios reproductores de vídeo en una sola instancia del componente FLVPlayback para reproducir varios vídeos y pasar de uno a otro durante la reproducción.

El reproductor de vídeo inicial se crea al arrastrar el componente FLVPlayback al escenario. El componente asigna automáticamente el número 0 al reproductor de vídeo inicial y lo convierte en el reproductor predeterminado. Para crear un reproductor de vídeo adicional, simplemente establezca la propiedad `activeVideoPlayerIndex` en un nuevo número. Cuando se establece la propiedad `activeVideoPlayerIndex`, se convierte además al reproductor de vídeo especificado en el reproductor de vídeo *activo*, que es el que se verá afectado por las propiedades y métodos de la clase FLVPlayback. Sin embargo, si se establece la propiedad `activeVideoPlayerIndex`, el reproductor de vídeo no pasará a ser visible. Para que el reproductor de vídeo sea *visible*, establezca la propiedad `visibleVideoPlayerIndex` en el número del reproductor de vídeo. Para más información sobre la interacción de estas propiedades con los métodos y propiedades de la clase FLVPlayback, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

El siguiente código ActionScript carga la propiedad `contentPath` para reproducir un archivo FLV en el reproductor de vídeo predeterminado y le añade un cuepoint. Cuando se produce el evento `ready`, el controlador de eventos abre un segundo reproductor de vídeo estableciendo la propiedad `activeVideoPlayerIndex` en el número 1. Especifica un archivo FLV y un cuepoint para el segundo reproductor de vídeo y, a continuación, convierte nuevamente al reproductor predeterminado (0) en el reproductor de vídeo activo.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
// añadir un cuepoint al reproductor predeterminado
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
my_FLVPlybk.addASCuePoint(3, "1st_switch");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // añadir un segundo reproductor de vídeo y crear un cuepoint para él
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  my_FLVPlybk.addASCuePoint(3, "2nd_switch");
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Para cambiar a otro archivo FLV mientras se reproduce uno, debe obtener el control para realizar el cambio en el código ActionScript. Los cuepoints permiten intervenir en puntos específicos del archivo FLV mediante un evento `cuePoint`. El siguiente código crea un detector del evento `cuePoint` y llama a una función de controlador que pausa el reproductor de vídeo activo (0), cambia al segundo reproductor (1) y reproduce su archivo FLV:

```
// crear un objeto detector
var listenerObject:Object = new Object();
// añadir una función de controlador para el evento cuePoint
listenerObject.cuePoint = function(eventObject:Object):Void {
    // mostrar el nº del reproductor de vídeo que provoca el evento
    trace("Hit cuePoint event for player: " + eventObject.vp);
    // probar el reproductor de vídeo y cambiar los archivos FLV de forma
  correspondiente
    if (eventObject.vp == 0) {
      my_FLVPlybk.pause(); // pausar el primer archivo FLV
      my_FLVPlybk.activeVideoPlayerIndex = 1; // establecer el segundo
  reproductor como activo
      my_FLVPlybk.visibleVideoPlayerIndex = 1; // establecer el segundo
  reproductor como visible
```

```
        my_FLVPlybk.play(); // iniciar reproducción del nuevo reproductor/
    archivo FLV
      } else if (eventObject.vp == 1) {
        my_FLVPlybk.pause(); // pausar el segundo archivo FLV
        my_FLVPlybk.activeVideoPlayerIndex = 0; // establecer el primer
    reproductor como activo
        my_FLVPlybk.visibleVideoPlayerIndex = 0; // establecer el primer
    reproductor como visible
        my_FLVPlybk.play(); // iniciar reproducción del primer reproductor
      }
}
// añadir detector de un evento cuePoint
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
    trace("Hit complete event for player: " + eventObject.vp);
    if (eventObject.vp == 0) {
      my_FLVPlybk.activeVideoPlayerIndex = 1;
      my_FLVPlybk.visibleVideoPlayerIndex = 1;
      my_FLVPlybk.play();
    } else {
      my_FLVPlybk.closeVideoPlayer(1);
    }
}
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Cuando se crea un nuevo reproductor de vídeo, la instancia de FLVPlayback establece sus propiedades en el valor del reproductor de vídeo predeterminado, excepto las propiedades `contentPath`, `totalTime` e `isLive`, que la instancia de FLVPlayback siempre establece en sus valores predeterminados: cadena vacía, 0 y `false`, respectivamente. Establece en `false` la propiedad `autoPlay`, que tiene el valor predeterminado `true` en el reproductor de vídeo predeterminado. La propiedad `cuePoints` no produce ningún efecto, ni siquiera si se carga posteriormente en el reproductor de vídeo predeterminado.

Los métodos y las propiedades que controlan el volumen, la posición, las dimensiones, la visibilidad y los controles de interfaz de usuario siempre son globales y su comportamiento *no* se ve afectado al establecer el valor de la propiedad `activeVideoPlayerIndex`. Para más información sobre estos métodos y propiedades, y sobre el efecto de establecer la propiedad `activeVideoPlayerIndex`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567. Las propiedades y métodos restantes se aplican al reproductor de vídeo identificado por el valor de la propiedad `activeVideoPlayerIndex`.

Sin embargo, las propiedades y métodos que controlan las dimensiones *sí interactúan* con la propiedad `visibleVideoPlayerIndex`. Para más información, consulte “FLVPlayback.visibleVideoPlayerIndex” en la página 711.

Flujo de archivos FLV de un servidor FCS

Si utiliza un servidor FCS para transmitir archivos FLV al componente FLVPlayback, debe añadir el archivo main.asc a la aplicación Flash Communication Server. El archivo main.asc se encuentra en la carpeta de la aplicación Flash 8/Samples and Tutorials/Samples/Components/FLVPlayback/main.asc.

Para configurar el servidor FCS para que transmita archivos FLV:

1. Cree una carpeta en la carpeta de la aplicación FCS y asígnele un nombre como **my_application**.
2. Copie el archivo main.asc en la carpeta my_application.
3. Cree una carpeta denominada **streams** en la carpeta my_application.
4. Cree una carpeta denominada **_definst_** en la carpeta streams.
5. Coloque los archivos FLV en la carpeta **_definst_**.

Para acceder a los archivos FLV de Flash Communication Server, utilice una URL como rtmp://*my_servername/my_application/stream.*flv.

Para más información sobre la administración de Flash Communication Server, incluida la configuración de un flujo en vivo, consulte la documentación de FCS en www.macromedia.com/support/documentation/en/flashcom/. Cuando se reproduce un flujo en vivo con FCS, es necesario establecer la propiedad `isLive` de FLVPlayback en `true`. Para más información, consulte `FLVPlayback.isLive` en la página 621.

Personalización del componente FLVPlayback

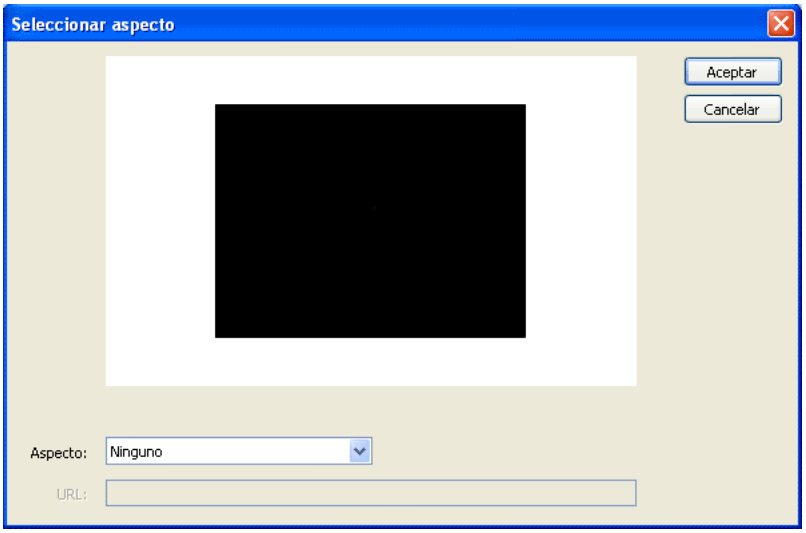
En esta sección se explica la forma de personalizar el componente FLVPlayback. Para ver información general detallada sobre cómo personalizar componentes, incluidos la terminología y los conceptos básicos sobre trabajo con estilos, aspectos y temas, consulte "Personalización de componentes" en *Utilización de componentes*. No obstante, la mayoría de los métodos que se utilizan para personalizar otros componentes no funcionan con el componente FLVPlayback. Para personalizar el componente FLVPlayback, utilice únicamente las técnicas que se describen en esta sección.

Tiene las siguientes opciones para personalizar el componente FLVPlayback: seleccionar un aspecto prediseñado, aplicar aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV o crear un nuevo aspecto. También puede utilizar las propiedades de FLVPlayback para modificar el comportamiento de un aspecto.

NOTA	Debe cargar el archivo SWF de aspecto en el servidor Web, junto con el archivo SWF de la aplicación para que el aspecto funcione con el componente FLVPlayback.

Selección de un aspecto prediseñado

Puede elegir un aspecto para el componente FLVPlayback haciendo clic en la celda `value` del parámetro `skin` en el inspector de componentes. A continuación, haga clic en el icono de lupa para abrir el siguiente cuadro de diálogo Seleccionar aspecto, que le permite seleccionar un aspecto o proporcionar una URL que especifique la ubicación del archivo SWF de aspecto.



Los aspectos que se enumeran en el menú emergente Aspecto se encuentran en la carpeta Configuration/Skins de Flash 8 o en la carpeta Configuration/Skins local del usuario. Para que los nuevos aspectos estén disponibles en este cuadro de diálogo deberá colocar sus archivos SWF en la carpeta Skins después de crearlos. El nombre aparece en el menú emergente con una extensión .swf. Para más información sobre cómo crear un conjunto de aspectos, consulte "Creación de un aspecto nuevo" en la página 550.

Si desea aplicar aspectos al componente FLVPlayback mediante los componentes de interfaz de usuario personalizados de reproducción FLV, seleccione Ninguno en el menú emergente.

Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV

Los componentes de interfaz de usuario personalizados de reproducción FLV permiten personalizar la apariencia de los controles de FLVPlayback en el archivo FLA y permiten ver los resultados en una previsualización de la página Web. Sin embargo, estos componentes no están diseñados para cambiar de tamaño. Debería editar un clip de película y su contenido con un tamaño específico. Por este motivo, es preferible tener el componente FLVPlayback en el escenario con el tamaño deseado, con las propiedades `autoSize` y `maintainAspectRatio` establecidas en `false`.

Para empezar, arrastre simplemente los componentes de interfaz de usuario personalizados de reproducción FLV que desee desde el inspector de componentes hasta el escenario y asigne a cada uno de ellos un nombre de instancia en el inspector de propiedades. Cuando los componentes estén en el escenario, podrá editarlos como haría con cualquier otro símbolo.

Cuando abra los componentes observará que la configuración de cada uno de ellos es ligeramente distinta.

Componentes Button

Los componentes Button tienen una estructura similar. Los distintos tipos de botón son: BackButton, ForwardButton, MuteButton, PauseButton, PlayButton, PlayPauseButton y StopButton. La mayoría tiene un solo clip de película en el fotograma 1, con el nombre de instancia placeholder_mc. Suele ser una instancia del estado normal del botón, aunque no siempre es así. El fotograma 2 contiene cuatro clips de película en el escenario para cada uno de los estados de visualización: normal, puntero encima, presionado y desactivado. (En tiempo de ejecución, el componente nunca pasa realmente al fotograma 2; estos clips de película se colocan aquí para facilitar la edición y para obligarles a que se carguen en el archivo SWF sin necesidad de activar la casilla de verificación Exportar en primer fotograma en el cuadro de diálogo Propiedades de símbolo. No obstante, debe activar la opción Exportar para ActionScript.)

Para aplicar aspectos al botón, edite cada uno de estos clips de película. Puede cambiar su tamaño y su apariencia.

Normalmente se muestra parte del código ActionScript en el fotograma 1. No es necesario modificar este script. Simplemente detiene la cabeza lectora en el fotograma 1 y especifica los clips de película que se utilizarán en cada estado.

Botones PlayPauseButton y MuteButton

Los botones PlayPauseButton y MuteButton se configuran de forma distinta al resto de botones; sólo tienen un fotograma con dos capas y ningún script. Dicho fotograma contiene dos botones, uno encima del otro. En el primer caso, un botón de reproducción y un botón de pausa; en el segundo caso, un botón para activar el silencio y otro para desactivarlo. Para aplicar aspectos a los botones PlayPauseButton o MuteButton, aplique aspectos a cada uno de estos dos botones internos tal y como se describe en "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543; no se requiere ninguna acción adicional.

Botones BackButton y ForwardButton

Los botones BackButton y ForwardButton también se configuran de forma distinta al resto de botones. En el fotograma 2, disponen de clips de película adicionales que pueden utilizarse como fotogramas alrededor de uno de los botones o de ambos. Estos clips de película no son necesarios ni ofrecen ninguna función especial y simplemente se proporcionan como una opción más. Para utilizarlos, arrástrelos al escenario desde el panel Biblioteca y colóquelos en el lugar que elija. Si no le interesan, no los utilice o elimínelos del panel Biblioteca.

La mayoría de los botones proporcionados se basan en un conjunto común de clips de película, de forma que es posible cambiar la apariencia de todos los botones de una sola vez. Puede utilizar esta capacidad o puede reemplazar los clips de película comunes y dotar a cada botón de un aspecto distinto.

Componente BufferingBar

El componente de barra de almacenamiento en búfer es sencillo: Está formado por una animación que se hace visible cuando el componente entra en estado de almacenamiento en búfer y no requiere ningún código de ActionScript especial para configurarlo. De forma predeterminada, es una barra discontinua que se mueve de izquierda a derecha y tiene aplicada una máscara rectangular que le confiere un aspecto de barra móvil. Esta configuración no presenta ninguna característica especial.

Aunque las barras de almacenamiento en búfer de los archivos SWF de aspecto utilizan la escala de 9 divisiones porque deben ajustar su escala en tiempo de ejecución, el componente de interfaz de usuario personalizado BufferingBar *no puede* utilizar la escala de 9 divisiones porque tiene clips de película anidados. Si desea alargar o ensanchar el componente BufferingBar, quizá deba cambiar su contenido en lugar de ajustar su escala.

Componentes SeekBar y VolumeBar

Los componentes SeekBar y VolumeBar son similares, aunque tienen distintas funciones. Cada uno de ellos tiene selectores, utiliza los mismos mecanismos de seguimiento de selectores y admite clips anidados para realizar un seguimiento del progreso y del grado de relleno.

En muchas partes del código ActionScript en el componente FLVPlayback se supone que el punto de registro del componente SeekBar o VolumeBar está situado en la esquina superior izquierda del contenido, de modo que es importante mantener esta convención. De lo contrario, podría tener problemas con los selectores y con los clips de película de progreso y grado de relleno.

Aunque las barras de búsqueda de los archivos SWF de aspecto utilizan la escala de 9 divisiones porque deben ajustar su escala en tiempo de ejecución, el componente de interfaz de usuario personalizado SeekBar *no puede* utilizar la escala de 9 divisiones porque tiene clips de película anidados. Si desea alargar o ensanchar el componente SeekBar, quizá deba cambiar su contenido en lugar de ajustar su escala.

Selector

El fotograma 2 contiene una instancia del clip de película de selector. Al igual que los componentes BackButton y ForwardButton, el componente nunca pasa realmente al fotograma 2; estos clips de película se colocan aquí para facilitar la edición y para obligarles a que se carguen en el archivo SWF sin necesidad de activar la casilla de verificación Exportar en primer fotograma en el cuadro de diálogo Propiedades de símbolo. No obstante, debe activar la opción Exportar para ActionScript.

Quizá haya observado que el clip de película del selector tiene un rectángulo en el fondo, cuyo alfa está establecido en 0. Este rectángulo aumenta el tamaño del área activa del selector para poder agarrarlo con mayor facilidad sin cambiar su aspecto, de forma similar al estado Zona activa de un botón. Dado que el selector se crea dinámicamente en tiempo de ejecución, debe ser un clip de película y no un botón. Este rectángulo con el alfa establecido en 0 no es necesario por ningún otro motivo y, normalmente, puede reemplazar el interior del selector por la imagen que desee. Sin embargo, es recomendable mantener el punto de registro centrado horizontalmente en mitad del clip de película del selector.

El siguiente código ActionScript del fotograma 1 del componente SeekBar controla el selector:

```
stop();
handleLinkageID = "SeekBarHandle";
handleLeftMargin = 2;
handleRightMargin = 2;
handleY = 11;
```

La llamada a la función `stop()` es necesaria debido al contenido del fotograma 2.

La segunda línea especifica el símbolo que se utilizará como selector. No es necesario cambiarlo si simplemente se edita la instancia del clip de película del selector en el fotograma 2. En tiempo de ejecución, el componente FLVPlayback crea una instancia del clip de película especificado en el escenario como elemento del mismo nivel de la instancia del componente Bar, lo que significa que tienen el mismo clip de película principal. De esta forma, si la barra se encuentra en el nivel raíz, el selector debe estar también en el nivel raíz.

La variable `handleLeftMargin` determina la ubicación original del selector (0%) y la variable `handleRightMargin` determina su ubicación final (100%). Los números indican los desplazamientos desde los extremos izquierdo y derecho del control de la barra, con números positivos que marcan los límites dentro de la barra y números negativos que marcan los límites fuera de ella. Estos desplazamientos especifican el lugar donde puede dirigirse el selector, en función de su punto de registro. Si coloca el punto de registro en mitad del selector, los extremos izquierdo y derecho del selector traspasarán los límites. Un clip de película de barra de búsqueda debe tener el punto de registro en la esquina superior izquierda de su contenido para que funcione correctamente.

La variable `handleY` determina la posición `y` del selector con respecto a la instancia de la barra. Esto se basa en los puntos de registro de cada clip de película. El punto de registro en el selector de ejemplo se encuentra en la punta del triángulo, lo que permite colocarlo con respecto a la parte visible, sin tener en cuenta el rectángulo de estado de zona activa invisible. Además, un clip de película de barra debe tener el punto de registro en la esquina superior izquierda de su contenido para que funcione correctamente.

De esta forma, con estos límites, si el control de la barra se establece en (100, 100) y tiene una anchura de 100 píxeles, el selector abarcará de 102 a 198 horizontalmente y se mantendrá en 111 verticalmente. Si cambia los valores de `handleLeftMargin` y `handleRightMargin` a -2 y de `handleY` a -11, el selector abarcará de 98 a 202 horizontalmente y se mantendrá en 89 verticalmente.

Clips de película de progreso y grado de relleno

El componente SeekBar tiene un clip de película de *progreso* y el componente VolumeBar tiene un clip de película de *grado de relleno* pero, en la práctica, los componentes SeekBar o VolumeBar pueden tener o no alguno de estos componentes, o ambos. Ambos componentes presentan la misma estructura y un comportamiento similar, pero realizan un seguimiento de distintos valores. Un clip de película de progreso se rellena a medida que se descarga el archivo FLV (lo que resulta útil sólo en las descargas de HTTP, porque siempre aparece lleno en las transmisiones desde servidores FCS) y un clip de película de grado de relleno se va rellenando a medida que el selector se mueve de izquierda a derecha.

El componente FLVPlayback busca estas instancias de clip de película a través de un nombre de instancia específico, de modo que la instancia del clip de película de progreso debe tener el clip de película de barra como elemento principal y el nombre de instancia progress_mc. La instancia del clip de película de grado de relleno debe tener el nombre de instancia fullness_mc.

Puede establecer clips de película de progreso y de grado de relleno con o sin la instancia del clip de película fill_mc anidada en ellos. El clip de película fullness_mc de VolumeBar muestra el método *con* el clip de película fill_mc y el clip de película progress_mc de SeekBar muestra el método *sin* el clip de película fill_mc.

El método con el clip de película fill_mc anidado resulta útil si se desea utilizar un relleno cuya escala no puede ajustarse sin distorsionar la apariencia.

En el clip de película fullness_mc de VolumeBar, se aplica una máscara a la instancia del clip de película fill_mc anidado. Puede aplicarle una máscara cuando cree el clip de película o esperar a que se cree la máscara dinámicamente en tiempo de ejecución. Si le aplica una máscara con un clip de película, denomine a la instancia **mask_mc** y configúrela de forma que fill_mc aparezca tal y como lo haría con un porcentaje del 100%. Si no aplica una máscara a fill_mc, la máscara creada dinámicamente será rectangular y tendrá el mismo tamaño que fill_mc al 100%.

Hay dos formas de mostrar el clip de película fill_mc con la máscara, en función de si el valor de fill_mc.slideReveal es `true` o `false`.

Si fill_mc.slideReveal es `true`, fill_mc se mueve de izquierda a derecha para mostrarse a través de la máscara. Con un porcentaje del 0%, aparecerá en el extremo izquierdo y no se mostrará a través de la máscara. A medida que aumenta el porcentaje, se va moviendo a la derecha hasta alcanzar el 100% y finalmente vuelve al lugar donde se creó en el escenario.

Si fill_mc.slideReveal es `false` o undefined (comportamiento predeterminado), se cambiará el tamaño de la máscara de izquierda a derecha para mostrar algo más de fill_mc. Si el porcentaje es 0%, se cambia el tamaño de la máscara a 05 horizontalmente y, a medida que va aumentando el porcentaje, el valor de `_xscale` aumenta hasta que, al 100%, muestra fill_mc completamente. En este punto, el valor de `_xscale` no es necesariamente 100 porque es posible que se ajustara la escala de mask_mc cuando se creó.

El método sin fill_mc es más sencillo que el método con fill_mc, pero distorsiona horizontalmente el relleno. Si desea evitar dicha distorsión, debe utilizar fill_mc. La instancia progress_mc de SeekBar ilustra este método.

El clip de película de progreso o de grado de relleno ajusta su escala horizontalmente en función del porcentaje. Al 0%, el valor de `_xscale` de la instancia se establece en 0, haciéndolo invisible. A medida que aumenta el porcentaje, el valor de `_xscale` se ajusta hasta que, al llegar al 100%, el clip alcanza el mismo tamaño que tenía cuando se creó en el escenario. Nuevamente, en este punto, el valor de `_xscale` no es necesariamente 100 porque es posible que se ajustara la escala de la instancia del clip cuando se creó.

Conexión de los componentes de interfaz de usuario personalizados de reproducción FLV

Debe escribir código ActionScript para conectar los componentes de interfaz de usuario personalizados de reproducción FLV a la instancia del componente FLVPlayback. Primero, debe asignar un nombre a la instancia de FLVPlayback y utilizar código ActionScript para asignar las instancias del componente de interfaz de usuario personalizado de reproducción FLV a las propiedades de FLVPlayback correspondientes. En el siguiente ejemplo, la instancia de FLVPlayback es my_FLVPlybk, los nombres de propiedades de FLVPlayback se especifican después de los puntos (.) y las instancias de control de interfaz de usuario personalizado de reproducción FLV se sitúan a la derecha del signo igual (=):

```
// instancia de FLVPlayback = my_FLVPlybk
my_FLVPlybk.playButton = playbtn; // establecer propiedad playButton en
  playbtn, etc.
my_FLVPlybk.pauseButton = pausebtn;
my_FLVPlybk.playPauseButton = playpausebtn;
my_FLVPlybk.stopButton = stopbtn;
my_FLVPlybk.muteButton = mutebtn;
my_FLVPlybk.backButton = backbtn;
my_FLVPlybk.forwardButton = forbtn;
my_FLVPlybk.volumeBar = volbar;
my_FLVPlybk.seekBar = seekbar;
my_FLVPlybk.bufferingBar = bufbar;
```

Ejemplo

En los siguientes pasos se crean controles StopButton, PlayPauseButton, MuteButton y SeekBar personalizados:

1. Arrastre el componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**.
2. Establezca el parámetro `contentPath` en **http://www.helpexamples.com/flash/video/cuepoints.flv** a través del inspector de componentes.
3. Establezca el parámetro Skin en None.

4. Arrastre una instancia de StopButton, PlayPauseButton y MuteButton al escenario y colóquelas sobre la instancia de FLVPlayback, apilándolas verticalmente a la izquierda. Asigne a cada botón un nombre de instancia en el inspector de propiedades (por ejemplo, **my_stopbttn**, **my_plypausbttn** y **my_mutebttn**).
5. En el panel Biblioteca, abra la carpeta FLVPlayback Skins y, a continuación, la subcarpeta SquareButton.
6. Seleccione el clip de película SquareBgDown y haga doble clic en él para abrirlo en el escenario.
7. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh), elija Seleccionar todo en el menú y elimine el símbolo.
8. Seleccione la herramienta Óvalo, dibuje un óvalo en la misma ubicación y establezca el color azul como relleno #(0033FF).
9. En el inspector de propiedades, establezca una anchura (An.:) de 40 y una altura (Al.:) de 20. Establezca la coordenada x (X:) en 0,0 y la coordenada y (Y:) en 0,0.
10. Repita los pasos 6 a 8 para SquareBgNormal, pero cambie el color de relleno por amarillo (#FFFF00).
11. Repita los pasos 6 a 8 para SquareBgOver, pero cambie el color de relleno por verde (#006600).
12. Edite los clips de película de los diversos iconos de símbolos en los botones (PauseIcon, PlayIcon, MuteOnIcon, MuteOffIcon y StopIcon). Encontrará estos clips de película en el panel Biblioteca, en la carpeta FLV Playback Skins/*Etiqueta* Button/Assets, donde *Etiqueta* es el nombre del botón; por ejemplo, Play, Pause, etc. Siga estos pasos en cada uno de ellos:
    a. Elija la opción Seleccionar todo.
    b. Cambie el color por rojo (#FF0000).
    c. Ajuste la escala un 300%.
    d. Cambie la ubicación de X: del contenido por 7,0 para modificar la ubicación horizontal del icono en cada estado de botón.

NOTA	Al cambiar la ubicación de esta forma, se evita tener que abrir cada estado de botón y mover la instancia de clip de película del icono.

13. Haga clic en la flecha Atrás azul situada por encima de la línea de tiempo para volver a la escena 1, fotograma 1.
14. Arrastre un componente SeekBar al escenario y colóquelo en la esquina inferior derecha de la instancia de FLVPlayback.

15. En el panel Biblioteca, haga doble clic en el componente SeekBar para abrirlo en el escenario.
16. Ajuste la escala un 400%.
17. Seleccione el contorno y establezca el color rojo (#FF0000).
18. Haga doble clic en SeekBarProgress en la carpeta FLVPlayback Skins/Seek Bar y establezca el color amarillo (#FFFF00).
19. Haga doble clic en SeekBarHandle en la carpeta FLVPlayback Skins/Seek Bar y establezca el color rojo (#FF0000).
20. Haga clic en la flecha Atrás azul situada por encima de la línea de tiempo para volver a la escena 1, fotograma 1.
21. Seleccione la instancia de SeekBar en el escenario y asígnele el nombre de instancia **my_seekbar**.
22. En el panel Acciones del fotograma 1 de la línea de tiempo, añada una sentencia de importación para las clases de vídeo y asigne nombres de botón y de barra de búsqueda a las propiedades de FLVPlayback correspondientes, tal y como se muestra en el siguiente ejemplo:

```
import mx.video.*;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.playPausebttn = my_plypausbttn;
my_FLVPlybk.muteButton = my_mutebttn;
my_FLVPlybk.seekBar = my_seekbar;
```

23. Pulse Control+Intro para probar la película.

Creación de un aspecto nuevo

La mejor forma de crear un archivo SWF de aspecto es copiar uno de los archivos de aspecto que incorpora Flash 8 y utilizarlo como punto de partida. Encontrará los archivos FLA de estos aspectos en la carpeta de la aplicación Flash 8 <*idioma*>/Configuration/SkinFLA. Para que el archivo SWF de aspecto finalizado esté disponible como una opción en el cuadro de diálogo Seleccionar aspecto, colóquelo en la carpeta <*idioma*>/Configuration/Skins de la carpeta de la aplicación Flash 8 o en la carpeta Configuration/Skins local de un usuario.

Como verá, es muy sencillo realizar modificaciones para cambiar la apariencia de un botón o del fondo *cromático* de un botón sin necesidad de cambiar las dimensiones. Todos los aspectos instalados tienen los mismos botones basados en distintos fondos cromáticos, de forma que puede realizar cambios radicales simplemente cambiando el color del fondo cromático. Por ejemplo, puede reorganizar los controles del clip de película de diseño simplemente moviendo los clips de marcadores de posición. Puede ver estos cambios tal y como aparecerán en el archivo SWF finalizado.

Cuando observe los archivos FLA de aspectos de Flash 8 instalados, puede parecer que algunos elementos del escenario no son necesarios, pero muchos de estos elementos se incluyen en capas de guía. Para ver rápidamente lo que aparece realmente en el archivo SWF, escriba **Ctl-Enter** para probar la película. De esta forma verá también cómo afecta la escala de 9 divisiones a determinados controles, ya que la escala de 9 divisiones no aparece activa durante la edición.

En las siguientes secciones se describen personalizaciones y cambios más complejos de los clips de película SeekBar, BufferingBar y VolumeBar.

Utilización de layout_mc

Cuando abra un archivo FLA de aspectos de Flash 8, encontrará un clip de película denominado layout_mc en la esquina superior izquierda del escenario. El clip *debe* denominarse `l`ayout_mc. El clip layout_mc y el código ActionScript que encuentre en el mismo fotograma definen la disposición de los controles en tiempo de ejecución.

Aunque layout_mc se parezca mucho a la apariencia del aspecto en tiempo de ejecución, el contenido de este clip no es visible en tiempo de ejecución. Sólo se utiliza para calcular dónde se ubicarán los controles. Los otros controles del escenario se utilizarán en tiempo de ejecución.

En layout_mc hay un marcador de posición del componente FLVPlayback, denominado video_mc. Todos los demás controles se disponen con respecto a video_mc. Si empieza con uno de los archivos FLA de Flash 8 y cambia el tamaño de los controles, probablemente podrá ajustar el diseño moviendo estos clips de marcador de posición.

Cada uno de los clips de marcador de posición tiene un nombre de instancia específico. Los nombres de los clips de marcador de posición son: playpause_mc, play_mc, pause_mc, stop_mc, back_mc, bufferingBar_mc, seekBar_mc, volumeMute_mc y volumeBar_mc.

No importa qué clip se utilice para un control. Normalmente, en los botones se utiliza el clip de estado normal. En los demás controles, se utiliza el clip para dicho control, pero esto es sólo por comodidad. Lo único que importa es la ubicación de *x* (horizontal) e *y* (vertical) y la altura y anchura del marcador de posición.

Puede tener tantos clips de fondo y de primer plano como desee, además de los controles estándar. Sin embargo, debe utilizar la siguiente convención de asignación de nombres: bg1_mc, bg2_mc, etc. para los clips de fondo; y fg1_mc, fg2_mc, etc. para los clips de primer plano. No puede omitir ningún número. Por ejemplo, si tiene bg1_mc y bg3_mc pero no tiene bg2_mc, no se utilizará bg3_mc. Este esquema se ha diseñado para poner clips de fondo detrás de los controles, con bg1_mc en el fondo y bg2_mc por encima, y clips de primer plano encima de los controles, con fg1_mc primero y fg2_mc encima de fg1_mc, etc. Sin embargo, la relación de capas de los clips viene determinada realmente por el orden de los controles correspondientes en el escenario, de modo que asegúrese de que es correcto.

El clip bg1_mc es especial. Si establece la propiedad `FlvPlayback.skinAutoHide` en `true`, el aspecto muestra cuándo se desplaza el ratón por encima del clip bg1_mc. Es importante para los aspectos que aparecen fuera de los límites del reproductor de vídeo. Para más información sobre la propiedad `skinAutoHide`, consulte "Modificación del comportamiento de aspecto" en la página 557.

En los archivos FLA de Flash 8, bg1_mc se utiliza para el fondo cromático y en algunos de ellos se utiliza bg2_mc para el borde de los botones para avanzar y retroceder.

ActionScript

El siguiente código ActionScript se aplica de forma general a todos los controles. Algunos controles tienen código ActionScript específico que define un comportamiento adicional y que se explica en la sección correspondiente a dicho control.

El código ActionScript inicial define la anchura y la altura mínimas del aspecto. El cuadro de diálogo Seleccionar aspecto muestra estos valores, que se utilizan en tiempo de ejecución para evitar que el aspecto ajuste su escala por debajo del tamaño mínimo. Si no desea especificar un tamaño mínimo, déjelo como valor undefined, o inferior o igual a cero.

```
// anchura y altura mínimas de vídeo recomendadas para este aspecto,
// dejar como undefined o <= 0 si no hay un mínimo
layout_mc.minWidth = 270;
layout_mc.minHeight = 60;
```

En cada marcador de posición pueden aplicarse las siguientes propiedades:

Propiedad	Descripción
`mc:MovieClip`	La instancia de este control en el escenario. Si no se establece, `layout_mc.foo_mc.mc` tiene el valor predeterminado `foo_mc`.
`anchorLeft:Boolean`	Ubica el control con respecto al lado izquierdo de la instancia de FLVPlayback. El valor predeterminado es `true` a menos que `anchorRight` se establezca explícitamente en `true`; luego el valor predeterminado pasa a ser `false`
`anchorRight:Boolean`	Ubica el control con respecto al lado derecho de la instancia de FLVPlayback. El valor predeterminado es `false`.
`anchorBottom:Boolean`	Ubica el control con respecto a la parte inferior de la instancia de FLVPlayback. El valor predeterminado es `true` a menos que `anchorTop` se establezca explícitamente en `true`; luego el valor predeterminado pasa a ser `false`.
`anchorTop:Boolean`	Ubica el control con respecto a la parte superior de la instancia de FLVPlayback. El valor predeterminado es `false`.

Si las propiedades `anchorLeft` y `anchorRight` tienen el valor `true`, el control ajustará su escala horizontalmente en tiempo de ejecución. Si las propiedades `anchorTop` y `anchorBottom` tienen el valor `true`, el control ajustará su escala verticalmente en tiempo de ejecución.

Para ver los efectos de estas propiedades, consulte su uso en los aspectos de Flash 8. Los controles BufferingBar y SeekBar son los únicos que ajustan su escala, y se disponen uno encima del otro y tienen las propiedades `anchorLeft` y `anchorRight` establecidas en `true`. Todos los controles a la izquierda de BufferingBar y SeekBar tienen la propiedad `anchorLeft` establecida en `true`, y todos los controles a la derecha tienen la propiedad `anchorRight` establecida en `true`. Todos los controles tienen la propiedad `anchorBottom` establecida en `true`.

Puede intentar editar el clip de película `layout_mc` para crear un aspecto donde los controles se sitúen en la parte superior en lugar de en la parte inferior. Simplemente necesita mover los controles a la parte superior, con respecto a `video_mc`, y establecer `anchorTop` en `true` en todos los controles.

Estados de botón

Todos los estados de botón se disponen en el escenario, pero no importa el lugar donde se coloquen estas instancias de clip de película en el escenario. Sin embargo, sí importa que se aniden en los clips de película de una forma específica y que cada instancia de clip de película tenga el nombre de instancia correcto.

La estructura de las instancias de clip y sus nombres de instancia se muestran en el siguiente ejemplo:

```
playpause_mc
  play_mc
    up_mc, over_mc, down_mc, disabled_mc
  pause_mc
    up_mc, over_mc, down_mc, disabled_mc
stop_mc
  up_mc, over_mc, down_mc, disabled_mc
back_mc
  up_mc, over_mc, down_mc, disabled_mc
forward_mc
  up_mc, over_mc, down_mc, disabled_mc
volumeMute_mc
  on_mc
    up_mc, over_mc, down_mc, disabled_mc
  off_mc
    up_mc, over_mc, down_mc, disabled_mc
```

Observe que los archivos FLA de Flash 8 tienen botones adicionales para avanzar y retroceder en el escenario. Se encuentran en capas de guía y muestran el uso de los clips de película ForwardBackBorder, ForwardBorder y BackBorder. Para más información, consulte “Clips de fondo y primer plano” en la página 556.

Puede editar los diversos estados como desee. Recuerde que todos los estados se ubican en el mismo lugar por medio de sus puntos de registro, de modo que si algunos estados son más grandes que otros, quizás no pueda colocar su trabajo en (0, 0), como ocurre en la mayoría de los aspectos de botón de Flash 8. Quizá resulte más sencillo, en algunos casos, mantener el punto de registro en el centro del trabajo.

Si no desea utilizar todos los estados, puede omitir algunos, pero debe conservar up_mc. El clip up_mc se utiliza en lugar de los estados que se omiten.

Si desea tener botones de reproducción y pausa independientes, en lugar de tener un botón combinado para pausa y reproducción, coloque los clips play_mc y pause_mc en el escenario sin ajustarlos con un clip playpause_mc.

No es necesario ningún código adicional para configurar los botones, además del código descrito en "Utilización de layout_mc" en la página 551.

Barra de almacenamiento en búfer

La barra de almacenamiento en búfer tiene dos clips de película: bufferingBar_mc y bufferingBarFill_mc. La posición de cada clip en el escenario con respecto al otro clip es importante porque se mantiene su posición relativa. La barra de almacenamiento en búfer utiliza dos clips independientes porque el componente ajusta la escala de bufferingBar_mc pero no de bufferingBarFill_mc.

El clip bufferingBar_mc tiene aplicada una escala de 9 divisiones, de forma que los bordes no se verán distorsionados al cambiar la escala. El clip bufferingBarFill_mc es muy amplio y no necesita ningún ajuste de escala. Se enmascara automáticamente en tiempo de ejecución para mostrar únicamente la parte situada por encima del clip bufferingBar_mc estirado. De forma predeterminada, las dimensiones exactas de la máscara mantendrán un margen igual a izquierda y derecha en el clip bufferingBar_mc, basándose en la diferencia entre las posiciones de *x* (horizontal) de bufferingBar_mc y _mc. Para personalizar la posición, utilice código ActionScript.

Si la barra de almacenamiento en búfer no necesita ajustar la escala o no utiliza escala en 9 divisiones, puede configurarla como el componente de interfaz de usuario personalizado de reproducción FLV BufferingBar. Para más información, consulte "Componente BufferingBar" en la página 544.

La barra de almacenamiento en búfer tiene la siguiente propiedad adicional:

Propiedad	Descripción
`fill_mc:MovieClip`	Especifica el nombre de instancia del relleno de la barra de almacenamiento en búfer. Su valor predeterminado es bufferingBarFill_mc.

Barra de búsqueda y barra de volumen

La barra de búsqueda también tiene dos clips de película: seekBar_mc y seekBarProgess_mc. La posición de cada clip en el escenario con respecto al otro clip es importante porque se mantiene su posición relativa. Aunque ambos clips ajustan su escala, el clip seekBarProgress_mc no puede anidarse en seekBar_mc porque seekBar_mc utiliza la escala en 9 divisiones, que no funciona correctamente con clips de película anidados.

El clip seekBar_mc tiene aplicada una escala de 9 divisiones, de forma que los bordes no se verán distorsionados al cambiar la escala. El clip seekBarProgress_mc también ajusta su escala pero no se distorsiona. No utiliza la escala de 9 divisiones porque es un relleno y no se ve afectado cuando se distorsiona.

El clip seekBarProgress_mc funciona sin fill_mc, de forma muy similar al modo en que el clip progress_mc funciona en los componentes de interfaz de usuario personalizados de reproducción de FLV. Dicho de otro modo, no se enmascara y ajusta su escala horizontalmente. Las dimensiones exactas del clip bufferingBarProgress_mc al 100% se definen por los márgenes izquierdo y derecho del clip bufferingBar_mc. Estas dimensiones son iguales, de forma predeterminada, y se basan en la diferencia entre las posiciones de *x* (horizontal) de seekBar_mc y seekBarProgress_mc. Puede personalizar las dimensiones mediante código ActionScript en el clip de película de la barra de búsqueda, tal y como se muestra en el siguiente ejemplo:

```
progressLeftMargin = 2;
progressRightMargin = 2;
progressY = 11;
fullnessLeftMargin = 2;
fullnessRightMargin = 2;
fullnessY = 11;
```

Al igual que el componente de interfaz de usuario personalizado de reproducción de FLV SeekBar, es posible crear un clip de película de grado de relleno para la barra de búsqueda. Si la barra de búsqueda no necesita un ajuste de escala o si se ajusta su escala pero no utiliza la escala de 9 divisiones, puede configurar el clip progress_mc o fullness_mc mediante cualquiera de los métodos que se utilizan para los componentes de interfaz de usuario personalizados de reproducción FLV. Para más información, consulte “Clips de película de progreso y grado de relleno” en la página 546.

Dado que la barra de volumen de aspectos de Flash 8 no ajusta su escala, se crea de la misma forma que el componente de interfaz de usuario personalizado de reproducción de FLV VolumeBar. Para más información, consulte “Componentes SeekBar y VolumeBar” en la página 545. La excepción es que el selector se implementa de forma distinta. Para más información, consulte la sección siguiente.

Selector

Los selectores de SeekBar y VolumeBar se colocan en el escenario, junto a la barra. De forma predeterminada, el margen izquierdo, el margen derecho y los valores del eje *y* del selector se establecen mediante su posición relativa al clip de película de la barra. El margen izquierdo se establece mediante la diferencia entre la posición *x* (horizontal) del selector y la posición *x* (horizontal) de la barra, y el margen derecho es igual al margen izquierdo. Puede personalizar estos valores mediante código ActionScript en el clip de película SeekBar o VolumeBar. El siguiente ejemplo es el mismo código ActionScript que se utiliza con los componentes de interfaz de usuario personalizados de reproducción de FLV:

```
handleLeftMargin = 2;
handleRightMargin = 2;
handleY = 11;
```

Más allá de estas propiedades, los selectores son simples clips de película configurados del mismo modo que en los componentes de interfaz de usuario personalizados de reproducción de FLV. Ambos tienen fondos rectangulares con la propiedad `alpha` establecida en 0. Estos fondos sólo sirven para aumentar la zona activa y no son necesarios.

ActionScript

La barra de búsqueda y la barra de volumen admiten las siguientes propiedades adicionales:

Propiedad	Descripción
`handle_mc:MovieClip`	Clip de película del selector. El valor predeterminado es seekBarHandle_mc o volumeBarHandle_mc
`progress_mc:MovieClip`	Clip de película de progreso. El valor predeterminado es seekBarProgress_mc o volumeBarProgress_mc
`fullness_mc:MovieClip`	Clip de película de grado de relleno. El valor predeterminado es seekBarFullness o volumeBarFullness.

Clips de fondo y primer plano

Los clips de película chrome_mc y forwardBackBorder_mc se implementan como clips de fondo.

De los clips de película ForwardBackBorder, ForwardBorder y BackBorder situados en el escenario y los botones de marcador de posición para avanzar y retroceder, el único que *no* es una capa de guía es ForwardBackBorder. Sólo se encuentra en los aspectos que utilizan realmente los botones para avanzar y retroceder.

No se requiere ningún código ActionScript adicional para configurar los clips de fondo y de primer plano.

Modificación del comportamiento de aspecto

La propiedad `bufferingBarHidesAndDisablesOthers` y la propiedad `skinAutoHide` permiten personalizar el comportamiento del aspecto de FLVPlayback.

Si se establece la propiedad `bufferingBarHidesAndDisablesOthers` en `true`, el componente FLVPlayback oculta SeekBar y su selector, y además desactiva los botones de reproducción y pausa cuando el componente pasa al estado de almacenamiento en búfer. Esto puede ser útil cuando se transmite un archivo FLV desde FCS a través de una conexión lenta con un valor alto establecido en la propiedad `bufferTime` (10, por ejemplo). En esta situación, un usuario impaciente podría intentar iniciar la búsqueda haciendo clic en los botones de reproducción y pausa, lo que podría demorar incluso más la reproducción del archivo. Para evitarlo, puede establecer `bufferingBarHidesAndDisablesOthers` en `true` y desactivar el elemento SeekBar y los botones de reproducción y pausa mientras el componente está en el estado de almacenamiento en búfer.

La propiedad `skinAutoHide` sólo afecta a archivos SWF de aspectos prediseñados y no a los controles creados desde componentes de interfaz de usuario personalizados de reproducción de FLV. Si se establece en `true`, el componente FLVPlayback oculta el aspecto cuando el puntero del ratón no está sobre el área de visualización. El valor predeterminado de esta propiedad es `true`.

Clase FLVPlayback

Herencia MovieClip > Clase FLVPlayback

Nombre de clase de ActionScript `mx.video.FLVPlayback`

FLVPlayback amplía la clase MovieClip y ajusta un objeto VideoPlayer. Para más información sobre la clase VideoPlayer, consulte "Clase VideoPlayer" en la página 729.

A diferencia de otros componentes, el componente FLVPlayback no amplía UIObject ni UIComponent y, por lo tanto, no admite los métodos y propiedades de estas clases. Por ejemplo, debe llamar al método `attachMovie()` de MovieClip en lugar de llamar al método `createClassObject()` de UIObject para crear una instancia del componente en ActionScript.

Los métodos y las propiedades de la clase FLVPlayback permiten reproducir y manipular archivos FLV en la aplicación Flash mediante el componente FLVPlayback.

Si se establece una propiedad de la clase FLVPLayback con ActionScript, se sustituye en el inspector de propiedades o el inspector de componentes un parámetro equivalente que establece inicialmente el valor de la propiedad.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. El siguiente código muestra la versión en el panel Salida:

```
trace(mx.video.FLVPlayback.version);
```

Resumen de métodos de la clase FLVPlayback

En la tabla siguiente se enumeran los métodos de la clase FLVPlayback:

Método	Descripción
`FLVPlayback.addASCuePoint()`	Añade un cuepoint de ActionScript.
`FLVPlayback.addEventListener()`	Crea un detector de un evento especificado.
`FLVPlayback.bringVideoPlayerToFront()`	Trae un reproductor de vídeo al frente de la pila de reproductores de vídeo.
`FLVPlayback.closeVideoPlayer()`	Cierra NetStream en el reproductor de vídeo con el índice especificado y elimina el reproductor de vídeo.
`FLVPlayback.findCuePoint()`	Busca el tipo de cuepoint especificado que tiene el tiempo y el nombre especificados, o al menos uno de ellos.
`FLVPlayback.findNearestCuePoint()`	Busca el tipo de cuepoint especificado en el tiempo especificado (o cerca del mismo) o con el nombre especificado.
`FLVPlayback.findNextCuePointWithName()`	Busca el siguiente cuepoint con el mismo nombre que un cuepoint devuelto por los métodos `findCuePoint()` o `findNearestCuePoint()`.
`FLVPlayback.getVideoPlayer()`	Obtiene el reproductor de vídeo especificado por el parámetro *`index`*.
`FLVPlayback.isFLVCuePointEnabled()`	Devuelve `false` si el cuepoint incorporado del archivo FLV ha sido desactivado mediante ActionScript.
`FLVPlayback.load()`	Inicia la carga del archivo FLV con la propiedad `autoPlay` establecida en `false`.
`FLVPlayback.pause()`	Pausa la reproducción del flujo de vídeo.
`FLVPlayback.play()`	Inicia la reproducción del flujo de vídeo y permite cargar y reproducir un nuevo archivo FLV.

Método	Descripción
FLVPlayback.removeASCuePoint()	Elimina un cuepoint de ActionScript.
FLVPlayback.removeEventListener()	Elimina un detector de eventos.
FLVPlayback.seek()	Busca un tiempo especificado en el archivo (en segundos, con una precisión decimal de milisegundos).
FLVPlayback.seekPercent()	Busca un porcentaje del tiempo reproducido del archivo.
FLVPlayback.seekSeconds()	Igual que FLVPlayback.seek().
FLVPlayback.seekToNavCuePoint()	Busca el cuepoint de navegación con el nombre especificado en el tiempo especificado (o después de dicho tiempo).
FLVPlayback.seekToNextNavCuePoint()	Busca el siguiente cuepoint de navegación con respecto al tiempo especificado.
FLVPlayback.seekToPrevNavCuePoint()	Busca el anterior cuepoint de navegación con respecto al tiempo especificado.
FLVPlayback.setFLVCuePointEnabled()	Activa o desactiva uno o más cuepoints del archivo FLV.
FLVPlayback.setScale()	Establece simultáneamente los valores de scaleX y scaleY.
FLVPlayback.setSize()	Establece simultáneamente los valores de width y height.
FLVPlayback.stop()	Detiene la reproducción del flujo de vídeo.

Resumen de propiedades de la clase FLVPlayback

La clase FLVPlayback tiene propiedades tanto de clase como de instancia.

Propiedades de la clase FLVPlayback

Las siguientes propiedades sólo se aplican a la clase FLVPlayback. Son constantes de sólo lectura que se aplican a todas las instancias del componente FLVPlayback en la aplicación.

Propiedad	Valor	Descripción
FLVPlayback.ACTIONSCRIPT	"actionscript"	Se puede utilizar como parámetro type para los métodos findCuePoint() y findNearestCuePoint().

Propiedad	Valor	Descripción
FLVPlayback.ALL	"all"	Se puede utilizar como parámetro type para los métodos findCuePoint() y findNearestCuePoint().
FLVPlayback.BUFFERING	"buffering"	Valor para probar la propiedad de estado.
FLVPlayback.CONNECTION_ERROR	"connectionError"	Valor para probar la propiedad de estado.
FLVPlayback.DISCONNECTED	"disconnected"	Valor para probar la propiedad de estado.
FLVPlayback.EVENT	"event"	Se puede utilizar como parámetro type para los métodos findCuePoint() y findNearestCuePoint().
FLVPlayback.FLV	"flv"	Se puede utilizar como parámetro type para los métodos findCuePoint() y findNearestCuePoint().
FLVPlayback.LOADING	"loading"	Valor para probar la propiedad de estado.
FLVPlayback.NAVIGATION	"navigation"	Se puede utilizar como parámetro type para los métodos findCuePoint() y findNearestCuePoint().
FLVPlayback.PAUSED	"paused"	Valor para probar la propiedad de estado.
FLVPlayback.PLAYING	"playing"	Valor para probar la propiedad de estado.
FLVPlayback.REWINDING	"rewinding"	Valor para probar la propiedad de estado.
FLVPlayback.SEEKING	"seeking"	Valor para probar la propiedad de estado.
FLVPlayback.STOPPED	"stopped"	Valor para probar la propiedad de estado.
FLVPlayback.version	Número de versión del componente	Determina la versión del componente FLVPlayback en uso.

Propiedades de instancia

En la siguiente tabla se enumeran las propiedades de instancia de la clase FLVPlayback. Este conjunto de propiedades se aplica a cada instancia de un componente FLVPlayback en la aplicación. Por ejemplo, si arrastra dos instancias del componente FLVPlayback al escenario, cada instancia tiene un conjunto de estas propiedades.

Propiedad	Descripción
`FLVPlayback.activeVideoPlayerIndex`	Número que especifica qué flujo de archivo FLV se ve afectado por otros métodos, propiedades y eventos. Utilice esta propiedad para administrar varios flujos de archivos FLV; el valor predeterminado es 0.
`FLVPlayback.autoPlay`	Valor booleano. Si es `true`, especifica que el componente reproduce el archivo FLV inmediatamente después de cargarlo. El valor predeterminado es `true`.
`FLVPlayback.autoRewind`	Valor booleano. Si es `true` hace que se rebobine el archivo FLV hasta el primer fotograma cuando se detenga la reproducción.
`FLVPlayback.autoSize`	Valor booleano. Si es `true`, hace que se ajuste automáticamente el tamaño del vídeo a las dimensiones de origen.
`FLVPlayback.backButton`	Objeto MovieClip que es el control backButton.
`FLVPlayback.bitrate`	Número que especifica el ancho de banda del usuario en bits por segundo. En algunos casos se utiliza para decidir qué archivo FLV debe reproducirse.
`FLVPlayback.buffering`	Valor booleano. Es `true` si el vídeo está en estado de almacenamiento en búfer. Sólo lectura.
`FLVPlayback.bufferingBar`	Objeto MovieClip que es el control bufferingBar.
`FLVPlayback.bufferingBarHidesAndDisablesOthers`	Afecta al comportamiento de los controles cuando el componente pasa al estado de almacenamiento en búfer.
`FLVPlayback.bufferTime`	Valor que especifica el número de segundos que se almacenarán en la memoria antes de que se inicie la reproducción de un flujo de vídeo.
`FLVPlayback.bytesLoaded`	Valor que indica el número de bytes descargados para una descarga HTTP. Sólo lectura.

Propiedad	Descripción
`FLVPlayback.bytesTotal`	Valor que especifica el número total de bytes descargados para una descarga HTTP. Sólo lectura.
`FLVPlayback.contentPath`	Cadena que especifica la URL de un archivo FLV que se va a cargar.
`FLVPlayback.cuePoints`	Matriz que describe objetos cuepoint de ActionScript y cuepoints del archivo FLV incorporados desactivados. Nunca debe utilizarse directamente con ActionScript. Sólo debe establecerse a través del cuadro de diálogo Cuepoints.
`FLVPlayback.forwardButton`	Objeto MovieClip que es el control ForwardButton.
`FLVPlayback.height`	Número que especifica la altura del vídeo en píxeles.
`FLVPlayback.idleTimeout`	Cantidad de tiempo en milisegundos antes de que Flash cierre una conexión inactiva al servidor FCS tras pausar o detener una reproducción.
`FLVPlayback.isLive`	Valor booleano. Es `true` si el flujo de vídeo es dinámico.
`FLVPlayback.isRTMP`	Valor booleano. Es `true` si el archivo FLV se transmite desde un servidor FCS o FVSS. Sólo lectura.
`FLVPlayback.maintainAspectRatio`	Valor booleano. Si es `true`, mantiene la proporción de vídeo.
`FLVPlayback.metadata`	Objeto que es un paquete de información de metadatos que se recibe de una llamada a la función callback `onMetaData()`, si está disponible. Sólo lectura.
`FLVPlayback.metadataLoaded`	Valor booleano. Es `true` si se ha encontrado y procesado un paquete de metadatos o si está claro que no se va a procesar. Sólo lectura.
`FLVPlayback.muteButton`	Objeto MovieClip que es el control MuteButton.
`FLVPlayback.ncMgr`	Objeto INCManager que proporciona acceso a una instancia de la clase que implementa INCManager. Sólo lectura.
`FLVPlayback.pauseButton`	Objeto MovieClip que es el control PauseButton.
`FLVPlayback.paused`	Valor booleano. Es `true` si el archivo FLV está en estado de pausa. Sólo lectura.
`FLVPlayback.playButton`	Objeto MovieClip que es el control PlayButton.

Propiedad	Descripción
FLVPlayback.playheadPercentage	Número que especifica el tiempo de cabeza lectora como un porcentaje de la duración total del archivo FLV.
FLVPlayback.playheadTime	Número que representa el tiempo o la posición actual (en segundos) de la cabeza lectora, que puede ser un valor fraccionario.
FLVPlayback.playheadUpdateInterval	Número que es la cantidad de tiempo en milisegundos entre cada evento `playheadUpdate`.
FLVPlayback.playing	Valor booleano. Es `true` si se reproduce el archivo FLV. Sólo lectura.
FLVPlayback.playPauseButton	Objeto MovieClip que es el control PlayPauseButton.
FLVPlayback.preferredHeight	Número que especifica la altura del archivo FLV de origen.
FLVPlayback.preferredWidth	Número que especifica la anchura del archivo FLV de origen.
FLVPlayback.progressInterval	Número que es la cantidad de tiempo en milisegundos entre cada evento progress.
FLVPlayback.scaleX	Número que especifica la escala horizontal.
FLVPlayback.scaleY	Número que especifica la escala vertical.
FLVPlayback.scrubbing	Valor booleano. Es `true` si el usuario está arrastrando el selector seekBar. Sólo lectura.
FLVPlayback.seekBar	Objeto MovieClip que es el control SeekBar.
FLVPlayback.seekBarInterval	Número que especifica la frecuencia en milisegundos con que se debe comprobar el selector seekBar durante el desplazamiento. El valor predeterminado es 250.
FLVPlayback.seekBarScrubTolerance	Número (porcentaje) que especifica hasta dónde puede mover un usuario el selector scrubBar antes de que se produzca una actualización. El valor se especifica como un porcentaje (entre 1 y 100).
FLVPlayback.seekToPrevOffset	Número de segundos que utiliza el método `seekToPrevNavCuePoint()` cuando compara su tiempo con el del cuepoint anterior.
FLVPlayback.skin	Cadena que especifica el nombre de un archivo SWF de aspecto.

Propiedad	Descripción
FLVPlayback.skinAutoHide	Valor booleano. Si es true, oculta el aspecto del componente cuando el ratón no está sobre el vídeo. El valor predeterminado es false.
FLVPlayback.state	Cadena que especifica el estado del componente. Se establece con los métodos load(), play(), stop(), pause() y seek(). Sólo lectura.
FLVPlayback.stateResponsive	Valor booleano. Es true si el estado es interactivo. Sólo lectura.
FLVPlayback.stopButton	Objeto MovieClip que es el control StopButton.
FLVPlayback.stopped	Valor booleano. Es true si el estado es detenido. Sólo lectura.
FLVPlayback.totalTime	Número que representa el tiempo total de reproducción del vídeo en segundos.
FLVPlayback.transform	Objeto que proporciona acceso directo a los métodos Sound.setTransform() y Sound.getTransform() para proporcionar mayor control del sonido.
FLVPlayback.visible	Valor booleano. Si es true, hace que el componente FLVPlayback sea visible.
FLVPlayback.visibleVideoPlayerIndex	Número que puede utilizar para administrar varios flujos de archivos FLV. Establece qué instancia de reproductor de vídeo es visible y audible. El valor predeterminado es 0.
FLVPlayback.volume	Número del intervalo 0 a 100 que indica el nivel del control de volumen.
FLVPlayback.volumeBar	Objeto MovieClip que es el control VolumeBar.
FLVPlayback.volumeBarInterval	Número que especifica la frecuencia en milisegundos con que se debe comprobar el selector volumeBar durante el desplazamiento. El valor predeterminado es 250.
FLVPlayback.volumeBarScrubTolerance	Número (porcentaje) que especifica hasta dónde puede mover un usuario el selector de la barra de volumen antes de que se produzca una actualización.
FLVPlayback.width	Número que especifica la anchura de la instancia del componente en píxeles.

Propiedad	Descripción
FLVPlayback.x	Número que especifica la ubicación horizontal del reproductor de vídeo en píxeles.
FLVPlayback.y	Número que especifica la ubicación vertical del reproductor de vídeo en píxeles.

Resumen de eventos de la clase FLVPlayback

En la tabla siguiente se enumeran los eventos de la clase FLVPlayback:

Evento	Descripción
FLVPlayback.buffering	Se distribuye cuando se pasa al estado de almacenamiento en búfer.
FLVPlayback.close	Se distribuye cuando se cierra `NetConnection` porque se agotó el tiempo de espera o mediante una llamada al método `close()`.
FLVPlayback.complete	Se distribuye cuando finaliza la reproducción al llegar al final del archivo FLV.
FLVPlayback.cuePoint	Se distribuye cuando se llega a un cuepoint.
FLVPlayback.fastForward	Se distribuye cuando se mueve hacia adelante la ubicación de la cabeza lectora mediante una llamada al método `seek()`.
FLVPlayback.metadata	Se distribuye la primera vez que se llega a los metadatos del archivo FLV.
FLVPlayback.paused	Se distribuye cuando se pasa al estado de pausa.
FLVPlayback.playheadUpdate	Se distribuye cada 0,25 segundos, de forma predeterminada, mientras se está reproduciendo el archivo FLV. Puede especificar la frecuencia mediante la propiedad `playheadUpdateInterval`.
FLVPlayback.playing	Se distribuye cuando se pasa al estado de reproducción.
FLVPlayback.progress	Se distribuye cada 0,25 segundos, desde que se llama al método `load()` hasta que se hayan cargado todos los bytes o se produzca un error de red. Puede especificar la frecuencia mediante la propiedad `progressInterval`.
FLVPlayback.ready	Se distribuye cuando el archivo FLV está cargado y preparado para mostrarse.
FLVPlayback.resize	Se distribuye cuando se cambia el tamaño del vídeo.

Evento	Descripción
FLVPlayback.rewind	Se distribuye cuando se mueve hacia atrás la ubicación de la cabeza lectora mediante una llamada a `seek()` o cuando finaliza la operación de rebobinado automático.
FLVPlayback.scrubFinish	Se distribuye cuando el usuario detiene el desplazamiento de la línea de tiempo con SeekBar.
FLVPlayback.scrubStart	Se distribuye cuando el usuario inicia el desplazamiento de la línea de tiempo con SeekBar.
FLVPlayback.seek	Se distribuye cuando se cambia la ubicación de la cabeza lectora mediante una llamada a `seek()` o mediante el control correspondiente.
FLVPlayback.skinError	Se distribuye cuando se produce un error al cargar un archivo SWF de aspecto.
FLVPlayback.skinLoaded	Se distribuye cuando se carga un archivo SWF de aspecto.
FLVPlayback.stateChange	Se distribuye cuando cambia el estado de reproducción.
FLVPlayback.stopped	Se distribuye cuando se pasa al estado detenido.
FLVPlayback.volumeUpdate	Se distribuye cuando se cambia el volumen mediante la propiedad `volume`.

FLVPlayback.ACTIONSCRIPT

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.ACTIONSCRIPT
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"actionscript"` para utilizarla como propiedad `type` con los métodos `findCuePoint()` y `findNearestCuePoint()`.

Ejemplo

En el siguiente ejemplo se utiliza la constante `ACTIONSCRIPT` para establecer la propiedad `type` del método `findCuePoint()`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  var cuePt:Object = new Object(); // crear objeto cuepoint
  cuePt.time = 2.444;
  cuePt.name = "ASCuePt1";
  my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
}
my_FLVPlybk.addEventListener("ready", listenerObject)
listenerObject.playing = function(eventObject:Object) {
    var rtn_obj:Object = new Object();
    if(rtn_obj = my_FLVPlybk.findCuePoint(2.444,
  FLVPlayback.ACTIONSCRIPT)){
      trace("Found cue point " + rtn_obj.name);
    }
  }
my_FLVPlybk.addEventListener("playing", listenerObject)
```

Véase también

`FLVPlayback.findCuePoint()`

FLVPlayback.activeVideoPlayerIndex

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.activeVideoPlayerIndex`

Descripción

Propiedad; número que especifica qué instancia del reproductor de vídeo se verá afectada por otras API. Utilice esta propiedad para administrar varios flujos de archivos FLV. El valor predeterminado es 0.

Esta propiedad no hace visible el reproductor de vídeo; utilice la propiedad `visibleVideoPlayerIndex` para dicho propósito.

Se crea un nuevo reproductor de vídeo la primera vez que se asigna un valor numérico a `activeVideoPlayerIndex`. Cuando se crea el nuevo reproductor de vídeo, sus propiedades se establecen en el valor del reproductor de vídeo predeterminado (`activeVideoPlayerIndex == 0`), salvo `contentPath`, `totalTime` y `isLive`, que siempre se establecen en sus valores predeterminados (cadena vacía, 0 y `false` respectivamente) y `autoPlay`, que siempre es `false` (el valor predeterminado es `true` sólo para el reproductor de vídeo predeterminado, es decir, `0`). La propiedad `cuePoints` no produce ningún efecto, ni siquiera si se carga posteriormente en el reproductor de vídeo predeterminado.

Las API que controlan el volumen, la posición, las dimensiones, la visibilidad y los controles de interfaz de usuario siempre son globales y su comportamiento *no* se ve afectado al establecer el valor de `activeVideoPlayerIndex`. Establecer la propiedad `activeVideoPlayerIndex` no afecta a los siguientes propiedades y métodos en concreto.

Propiedades y métodos no afectados por activeVideoPlayerIndex

`backButton`	`playPauseButton`	`skin`	`width`
`bufferingBar`	`scaleX`	`stopButton`	`x`
`bufferingBarHidesAndDisablesOthers`		`transform`	`y`
`forwardButton`	`scaleY`	`visible`	`setSize()`
`height`	`seekBar`	`volume`	`setScale()`
`muteButton`	`seekBarInterval`	`volumeBar`	
`pauseButton`	`seekBarScrubTolerance`	`volumeBarInterval`	
`playButton`	`seekToPrevOffset`	`volumeBarScrubTolerance`	

NOTA

La propiedad `visibleVideoPlayerIndex`, *no* la propiedad `activeVideoPlayerIndex`, determina qué reproductor de vídeo controla el aspecto.

Sin embargo, las API que controlan las dimensiones interactúan con la propiedad `visibleVideoPlayerIndex`. Para más información, consulte "FLVPlayback.visibleVideoPlayerIndex" en la página 711.

Las API restantes se destinan a un reproductor de vídeo específico que depende del valor de `activeVideoPlayerIndex`.

Al detectar eventos, obtendrá todos los eventos para todos los reproductores de vídeo. Para distinguir el reproductor de vídeo al que se destina el evento, utilice la propiedad `vp` del evento, que es un número que corresponde al valor establecido de `activeVideoPlayerIndex` y `visibleVideoPlayerIndex`. Todos los eventos tienen esta propiedad *salvo* `resize` y `volume`, que no son específicos de un reproductor de vídeo, sino globales para la instancia de FLVPlayback.

Por ejemplo, para cargar un segundo archivo FLV en segundo plano, establezca `activeVideoPlayerIndex` en 1 y llame al método `load()`. Cuando esté preparado para mostrar este archivo FLV y ocultar el otro, establezca `visibleVideoPlayerIndex` en 1.

Ejemplo

El siguiente ejemplo crea dos reproductores de vídeo para reproducir dos archivos FLV de forma consecutiva en una sola instancia del archivo FLV. Establece la propiedad `activeVideoPlayerIndex` para cambiar entre los reproductores de vídeo y sus respectivos archivos FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// especificar nombre y ubicación del archivo FLV correspondiente al
  reproductor predeterminado
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // añadir un segundo reproductor de vídeo y especificar el nombre y la
  ubicación de su archivo FLV
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // restablecer el reproductor de vídeo predeterminado, que reproduce su
  archivo FLV automáticamente
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
  // si complete hace referencia al segundo archivo FLV, convertir al
  predeterminado en activo y visible
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
  }
  else { // convertir al segundo reproductor en activo y visible y
  reproducir archivo FLV
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// añadir detector de evento complete
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Véase también

FLVPlayback.bringVideoPlayerToFront(), FLVPlayback.getVideoPlayer(), Clase VideoPlayer, FLVPlayback.visibleVideoPlayerIndex,

FLVPlayback.addASCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.addASCuePoint(cuePoint:Object)
my_FLVplybk.addASCuePoint(time:Number, name:String[, parameters:Object])
```

Parámetros

cuePoint Objeto con propiedades *name*:String y *time*:Number (en segundos) que describen el cuepoint. También puede tener una propiedad *parameters*:Object con pares nombre/valor. Puede tener *type*:String establecido en "actionscript". Si falta *type* o tiene otro valor, se establecerá automáticamente. Si el objeto no se ajusta a estas convenciones, el método emite un error VideoError.

time Número que representa el tiempo del nuevo cuepoint que se va a añadir. Si utiliza el parámetro *time*, debe seguirle el parámetro *name*.

name Cadena que especifica el nombre del cuepoint si envía un parámetro *time* en lugar del objeto *CuePoint*.

parameters Parámetros opcionales del cuepoint.

Valor devuelto

Una copia del objeto cuepoint que se añadió con las siguientes propiedades adicionales:

- `array` Matriz de cuepoints en los que se buscó. Debe tratar esta matriz como de sólo lectura, ya que si añade, elimina o edita los objetos que contiene podría provocar un funcionamiento incorrecto de los cuepoints.
- `index` Índice en la matriz del cuepoint devuelto.

Descripción

Método; añade un cuepoint de ActionScript y tiene el mismo efecto que añadir un cuepoint de ActionScript a través del cuadro de diálogo Cuepoints, salvo que se produce cuando se ejecuta una aplicación y no durante el desarrollo de la misma.

La información de cuepoint se elimina cuando se establece la propiedad `contentPath`. Para establecer los datos del cuepoint para el siguiente archivo FLV que se va a cargar, establezca primero el valor de la propiedad `contentPath`.

Se permite añadir varios objetos cuepoint de AS con el mismo nombre y el mismo tiempo. Cuando elimine cuepoints de ActionScript con el método `removeASCuePoint()`, se eliminarán todos los cuepoints con el mismo nombre y tiempo.

Ejemplo

En el ejemplo siguiente se añaden dos objetos cuepoint de ActionScript a un archivo FLV. El ejemplo añade el primero mediante un parámetro *CuePoint* y el segundo mediante los parámetros *time* y *name*. Cuando se llega a cada uno de los cuepoints durante la reproducción, un detector de eventos cuePoint muestra el valor de la propiedad `playheadTime` en un área de texto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
```

```
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_ta.visible = false;
// crear objeto cuepoint
var cuePt:Object = new Object(); // crear objeto cuepoint
cuePt.time = 2.444;
cuePt.name = "ASCuePt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
// añadir segundo objeto cuepoint de AS mediante los parámetros time y name
my_FLVPlybk.addASCuePoint(5, "ASCuePt2");
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_ta.text = "Elapsed time in seconds: " + my_FLVPlybk.playheadTime;
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

`FLVPlayback.findCuePoint()`, `FLVPlayback.removeASCuePoint()`

FLVPlayback.addEventListener()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización:

```
my_FLVPlybk.addEventListener(event:String, listener:Object):Void
my_FLVPlybk.addEventListener(event:String, listener:Function):Void
```

Parámetros

event Cadena que especifica el nombre del evento para el cual se registra un detector. Si el detector es un objeto, es también el nombre de la función del objeto detector que hay que llamar.

listener Nombre del objeto detector o función que se registra para el evento.

Valor devuelto

Ninguno.

Descripción

Método; registra un objeto detector o función para un evento especificado. Si el detector es un objeto, éste debe tener una función definida con el mismo nombre que el evento. Si el detector es una función, se llamará al nombre de la función para controlar el evento.

Ejemplo

El siguiente ejemplo detecta un evento `complete` y muestra un mensaje en el área de texto cuando se produce.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
Usage 1: listener object
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object(); // crear un objeto detector
listenerObject.complete = function(eventObject:Object):Void {
    my_ta.text = "That's All Folks!";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("complete", listenerObject);
```

```
Usage 2: listener function
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
function the_end(eventObject:Object):Void {
    my_ta.text = "That's All Folks!";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("complete", the_end);
```

Véase también

Resumen de eventos de la clase FLVPlayback,

FLVPlayback.removeEventListener(),

FLVPlayback.ALL

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`mx.video.FLVPlayback.ALL`

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"all"`. Puede utilizar esta propiedad como parámetro `type` para los métodos `findCuePoint()` y `findNearestCuePoint()`.

Ejemplo

En el siguiente ejemplo se busca entre todos los cuepoints uno denominado `point2` que tiene un tiempo de 7.748. En el ejemplo se muestran las propiedades `type` y `time` del cuepoint encontrado.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
// crear objeto cuepoint
var listenerObject = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var cuePt:Object = new Object(); // crear objeto cuepoint
  cuePt.name = "point2";
  cuePt.time = 7.748;
  if(cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.ALL)) //buscar
  objeto cuepoint
    trace("found a " + cuePt.type + " cue point at " + cuePt.time);
  else
    trace("cue point not found");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`FLVPlayback.findCuePoint()`

FLVPlayback.autoPlay

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.autoPlay

Descripción

Propiedad; valor booleano. Si se establece en `true`, hace que el archivo FLV se reproduzca inmediatamente cuando se establezca el valor de la propiedad `contentPath`. Si se establece en `false`, el componente esperará el comando de reproducción. Aunque `autoPlay` esté establecido en `false`, el componente cargará el contenido inmediatamente. El valor predeterminado es `true`.

Si establece la propiedad en `true` en el intervalo entre la carga de nuevos archivos FLV, no se producirá ningún efecto hasta que establezca `contentPath`.

Ejemplo

En el ejemplo siguiente se desactiva la reproducción del archivo FLV para mover la cabeza lectora un 30% en el tiempo de reproducción e iniciar la reproducción en ese punto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekPercent(30);
  my_FLVPlybk.play();
};
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`

FLVPlayback.autoRewind

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.autoRewind

Descripción

Propiedad; valor booleano. Si es `true`, hace que el archivo FLV se rebobine hasta el fotograma 1 cuando se detenga la reproducción porque el reproductor alcanzó el final del flujo o porque se llamó al método `stop()`. Esta propiedad no se utiliza para flujos dinámicos. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se establece la propiedad `autoRewind` en `false` para evitar que el archivo FLV se rebobine automáticamente cuando finalice la reproducción.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en la Biblioteca
*/
my_FLVPlybk.autoRewind = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

FLVPlayback.autoSize

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.autoSize

Descripción

Propiedad; valor booleano. Si es `true` hace que se ajuste automáticamente el tamaño del vídeo a las dimensiones del archivo FLV origen. Si se cambia esta propiedad de `false` a `true` después de cargar un archivo FLV, se inicia inmediatamente el cambio de tamaño automático. El valor predeterminado es `false`.

Ejemplo

El siguiente ejemplo muestra primero las dimensiones de origen del archivo FLV (`preferredWidth` y `preferredHeight`) cuando el archivo FLV está listo para reproducirse. A continuación, llama a `setSize()` para cambiar las dimensiones de la instancia de FLVPlayback, lo cual activa un evento `resize`. Luego establece la propiedad `autoSize` en `true` y activa otro evento `resize` que restaura el tamaño a las dimensiones del archivo FLV de origen. El controlador de eventos `resize` muestra las dimensiones de la instancia de FLVPlayback en el panel Salida después de cada evento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.maintainAspectRatio = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  trace("FLV width is: " + my_FLVPlybk.preferredWidth + " FLV height is: "
  + my_FLVPlybk.preferredHeight);
  my_FLVPlybk.setSize(400, 400);
  my_FLVPlybk.autoSize = true;
};
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.resize = function(eventObject:Object) {
  trace("my_FLVPlybk width is: " + my_FLVPlybk.width + ";
  my_FLVPlybk.height is: " + my_FLVPlybk.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.maintainAspectRatio`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.resize`

FLVPlayback.backButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.backButton`

Descripción

Propiedad; objeto MovieClip que es el control de reproducción BackButton. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Si se hace clic en el botón BackButton, se desencadena un evento `rewind`.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton` y `stopButton` para asociar controles individuales de interfaz de usuario personalizados de reproducción FLV a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de reproducción FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) y StopButton (**my_stopbttn**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayButton, PauseButton y
   StopButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.forwardButton`, `FLVPlayback.rewind`

FLVPlayback.bitrate

Help ID:

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.bitrate

Descripción

Propiedad; número que especifica los bits por segundo establecidos para la transferencia del archivo FLV.

Al realizar una descarga progresiva puede utilizar el formato SMIL, pero debe establecer la velocidad, ya que no hay detección automática.

Al transmitir vídeo desde un servidor Flash Communication Server (FCS), puede proporcionar un archivo SMIL que describa cómo cambiar entre varios flujos en función del ancho de banda. Un servidor FCS detecta automáticamente el ancho de banda y, en este caso, omite el valor `bitrate`, si está establecido.

Para más información sobre el uso de un archivo SMIL, consulte "Utilización de un archivo SMIL" en la página 735.

Ejemplo

En el siguiente ejemplo se comprueban dos botones de opción para determinar la velocidad que se utilizará al seleccionar un archivo FLV del archivo SMIL especificado. Las etiquetas `video` relevantes en el archivo SMIL se muestran en el siguiente código:

```
<switch>
  <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
  <video src="myvideo_isdn.flv" dur="3:00.1">
</switch>
```

En conexiones de baja velocidad, el código establece la propiedad `bitrate` para obligar a seleccionar el archivo FLV apropiado. En conexiones de mayor velocidad, aprovecha la detección automática de ancho de banda para transmitir desde el servidor FCS y no establece la velocidad.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente RadioButton y, a continuación, un componente Label al panel Biblioteca. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo SMIL.

```
/**
 Se requiere:
  - Componente FLVPlayback en la Biblioteca
  - Componente RadioButton en la Biblioteca
  - Componente Label en la Biblioteca
*/
import mx.video.*;
import mx.controls.*;
```

```
this.createClassObject(Label, "my_prompt", 10);
my_prompt.text = "Please indicate your connection speed: ";
this.createClassObject(RadioButton, "dialup", 20, {label:"Dialup modem (56
  KB)", groupName:"radioGroup"});
this.createClassObject(RadioButton, "isdn", 30, {label:"ISDN (128 KB)",
  groupName:"radioGroup"});
my_prompt.autoSize = "left";
dialup.setSize(200, 30);
isdn.setSize(200, 30);
// Colocar componentes RadioButton en el escenario.
my_prompt.move(my_FLVPlybk.x, my_FLVPlybk.y + my_FLVPlybk.height + 40);
dialup.move(my_FLVPlybk.x, my_prompt.y + my_prompt.height + 5);
isdn.move(my_FLVPlybk.x, dialup.y + 15);

// Crear un objeto detector
var rbListener:Object = new Object();
rbListener.click = function(eventObject:Object){
   if(dialup.selected) {   // para el módem
     my_FLVPlybk.bitrate = 56000;
   } // permitir detección automática de rdsi (o anchos de banda superiores)
}
// Añadir detector
radioGroup.addEventListener("click", rbListener);
my_FLVPlybk.contentPath = "http://www.someserver.com/video/sample.smil";
```

FLVPlayback.bringVideoPlayerToFront()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVPlybk.bringVideoPlayerToFront(*index*:Number)

Parámetros

index Número que representa el índice del reproductor de vídeo que se va a traer al frente

Descripción

Método; trae un reproductor de vídeo al frente de la pila de reproductores de vídeo. Es útil para transiciones personalizadas entre reproductores de vídeo. El orden de la pila es el mismo que en la propiedad `activeVideoPlayerIndex`: 0 se sitúa debajo del todo, 1 por encima de 0, 2 por encima de 1 y así sucesivamente.

Ejemplo

El siguiente ejemplo utiliza dos reproductores de vídeo para reproducir dos archivos FLV. Cuando se produce cada uno de los tres cuepoints del primer archivo FLV (cuepoints.flv), el ejemplo llama al método `bringVideoPlayerToFront()` para traer el otro archivo FLV al frente. Como el ejemplo establece la propiedad `_alpha` en 75 para el reproductor de vídeo número 1, el archivo FLV (plane_cuepoints) que se reproduce en dicho reproductor es transparente, de forma que los dos archivos FLV son visibles cuando se trae al frente dicho archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/cuepoints.flv");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
    if (eventObject.target.contentPath == "http://www.helpexamples.com/
  flash/video/cuepoints.flv") {
        //se activa cuando está listo el primer archivo FLV
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.load("http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv");
    } else {
        //se activa cuando está listo el segundo archivo FLV
        eventObject.target.activeVideoPlayerIndex = 0;
        eventObject.target.play();
        eventObject.target.activeVideoPlayerIndex = 1;
        eventObject.target.play();
        var layerOnTop:MovieClip = eventObject.target.getVideoPlayer(1);
        layerOnTop._alpha = 75;
        layerOnTop._visible = true;
    }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {

    //traer uno u otro al frente, en función del nombre de cuepoint
    if (eventObject.info.name == "point1") {
     trace(eventObject.info.name + " : 0 to front");
        eventObject.target.bringVideoPlayerToFront(1);
    } else if (eventObject.info.name == "point2") {
     trace(eventObject.info.name + " : 1 to front");
        eventObject.target.bringVideoPlayerToFront(0);
    }else if (eventObject.info.name == "point3") {
     trace(eventObject.info.name + " : 0 to front");
        eventObject.target.bringVideoPlayerToFront(1);
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.getVideoPlayer(), Clase VideoPlayer, FLVPlayback.visibleVideoPlayerIndex

FLVPlayback.buffering

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.buffering = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("buffering", listenerObject);
```

Descripción

Evento; se distribuye cuando la instancia de FLVPlayback pasa al estado de almacenamiento en búfer. La instancia de FLVPlayback suele pasar a este estado inmediatamente después de una llamada al método `play()` o cuando se hace clic en el control `Play`, antes de pasar al estado de reproducción. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Véase FLVPlayback.activeVideoPlayerIndex en la página 567 y FLVPlayback.visibleVideoPlayerIndex en la página 711.

La instancia de FLVPlayback también distribuye el evento `stateChange` cuando se inicia el almacenamiento en búfer.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `buffering`. Cuando se produce un evento `buffering`, el controlador de eventos llama al método `trace()` para mostrar los valores de las propiedades `state` y `vp`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.buffering = function(eventObject:Object) {
  trace("The state property has a value of " + eventObject.target.state);
  trace("The video player number is: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("buffering", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.state`, `FLVPlayback.removeEventListener()`

FLVPlayback.BUFFERING

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.BUFFERING
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"buffering"`. Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado de almacenamiento en búfer.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `FLVPlayback.BUFFERING` cuando el componente pasa al estado de almacenamiento en búfer.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.BUFFERING)
    trace(FLVPlayback.BUFFERING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.buffering

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.buffering`

Descripción

Propiedad; valor booleano. Es `true` si el vídeo está en estado de almacenamiento en búfer. Sólo lectura.

Ejemplo

El siguiente ejemplo crea un detector del evento `buffering` y muestra un mensaje en el panel Salida cuando se produce el evento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.buffering){
    trace("The video is buffering");
  }
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.buffering`, `FLVPlayback.BUFFERING`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.bufferingBar

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`my_FLVPlybk``.bufferingBar`

Descripción

Propiedad; objeto MovieClip que es el control de la barra de almacenamiento en búfer. Este control aparece cuando el archivo FLV está en estado de carga o de almacenamiento en búfer. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se asocian controles de interfaz de usuario personalizados de reproducción FLV individuales a un componente FLVPlayback estableciendo las siguientes propiedades: `playPauseButton`, `stopButton`, `backButton`, `forwardButton` y `bufferingBar`. La barra de almacenamiento en búfer sólo aparece mientras el archivo FLV se almacena en búfer, antes de que se inicie la reproducción.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de reproducción FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: PlayPauseButton (**my_bkbttn**), StopButton (**my_stopbttn**), BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**) y BufferingBar (**my_buffrgbar**). Añada las siguientes líneas de código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes de interfaz de usuario personalizados de reproducción FLV
  PlayPauseButton, StopButton, BackButton y ForwardButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.bufferingBar = my_buffrgbar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.bufferingBarHidesAndDisablesOthers`

FLVPlayback.bufferingBarHidesAndDisablesOthers

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVPlybk.bufferingBarHidesAndDisablesOthers

Descripción

Propiedad; si se establece en `true`, oculta el control SeekBar y desactiva los controles Play, Pause, PlayPause, BackButton y ForwardButton mientras el archivo FLV está en estado de almacenamiento en búfer. Esto puede ser útil para evitar que un usuario utilice estos controles para intentar acelerar la reproducción del archivo FLV cuando se descarga o transmite a través de una conexión lenta.

Ejemplo

En el siguiente ejemplo se supone que se reproduce un archivo FLV transmitido desde un servidor FCS o FVSS. Se establece la propiedad `bufferingBarHidesAndDisablesOthers` para desactivar los controles Play, Pause, PlayPause, BackButton y ForwardButton, y para ocultar el control SeekBar mientras el archivo FLV se almacena en búfer.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo FLV en el servidor FCS.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 15;
my_FLVPlybk.bufferingBarHidesAndDisablesOthers = true;
my_FLVPlybk.contentPath = "rtmp://host_name/somefolder/vid_name.flv";
```

Véase también

`FLVPlayback.bufferingBar`

FLVPlayback.bufferTime

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.bufferTime

Descripción

Propiedad; número que especifica el número de segundos que se almacenarán en la memoria antes de que se inicie la reproducción de un flujo de vídeo. En los archivos FLV que se transmiten a través de RTMP, que no se descargan y sólo se almacenan en búfer, puede ser importante aumentar el valor predeterminado de 0,1. En los archivos FLV que se descargan progresivamente a través de HTTP, aumentar este valor no supone una gran ventaja, aunque sí puede mejorar la visualización de vídeo de alta calidad en un equipo antiguo que sea más lento.

NOTA	Esta propiedad no especifica la cantidad de archivo FLV que se descarga antes de iniciar la reproducción.

Ejemplo

En el siguiente ejemplo se establece un tiempo de búfer de 8 segundos para un archivo FLV transmitido desde un servidor FCS.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código el fotograma 1 de la línea de tiempo, en el panel Acciones. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo FLV en el servidor FCS.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 8;
my_FLVPlybk.contentPath = "rtmp://host_name/somefolder/vid_name.flv";
```

FLVPlayback.bytesLoaded

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.bytesLoaded

Descripción

Propiedad; número que indica el número de bytes descargados para una descarga HTTP. Devuelve –1 cuando no hay flujo, cuando el flujo procede de un servidor FCS o cuando la información no está disponible todavía. El valor devuelto sólo es útil para una descarga HTTP. Sólo lectura.

Ejemplo

En el ejemplo siguiente se muestra el valor inicial de la propiedad `bytesLoaded` y su valor cuando se produce evento `ready`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  trace("State is " + eventObject.state + "; ready to play");
  // mostrar el nº de bytes cargados en este punto
  trace("Bytes loaded: " + my_FLVPlybk.bytesLoaded);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
trace("Bytes loaded: " + my_FLVPlybk.bytesLoaded); // -1 si la carga no ha
  empezado
```

FLVPlayback.bytesTotal

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.bytesTotal

Descripción

Propiedad; número que especifica el número total de bytes descargados para una descarga HTTP. Devuelve -1 cuando no hay flujo, cuando el flujo procede de un servidor FCS o cuando la información no está disponible todavía. El valor devuelto sólo es útil para una descarga HTTP. Sólo lectura.

Ejemplo

En el siguiente ejemplo se utiliza la propiedad `bytesTotal` para mostrar el número de bytes que se cargan para una descarga HTTP. Cuando se produce el evento `metadataReceived`, el controlador de eventos muestra este valor en el área de texto my_ta.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código el fotograma 1 de la línea de tiempo, en el panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
my_ta.visible = false;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
    my_ta.text = "Loading: " + my_FLVPlybk.bytesTotal + " bytes.";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

FLVPlayback.close

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("close", listenerObject);
```

Descripción

Evento; la instancia de FLVPlayback distribuye este evento cuando cierra NetConnection porque se ha agotado el tiempo de espera o se ha realizado una llamada al método `closeVideoPlayer()`, o porque se ha llamado al método `load()` o `play()`, o se ha establecido `contentPath`, provocando que se cierre la conexión RTMP. La instancia de FLVPlayback sólo distribuye este evento al transmitir desde un servidor FCS o FVSS. El objeto de evento tiene las propiedades `state` y `playheadTime`.

El evento tiene la propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento.

Ejemplo

En el siguiente ejemplo se supone que se reproduce un archivo FLV transmitido desde un servidor FCS o FVSS. Cuando finaliza el archivo FLV, un detector del evento `complete` establece la propiedad `contentPath` en la ubicación de un nuevo archivo FLV, lo que activa un evento `close` en la conexión RTMP para el primer archivo FLV. El detector del evento `close` muestra el número de índice del reproductor de vídeo afectado por el evento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo FLV en el servidor FCS.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
```

```
// detectar evento close en conexión RTMP; mostrar índice del reproductor de
  vídeo
listenerObject.close = function(eventObject:Object) {
  trace("Closed connection for video player: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("close", listenerObject);
// detectar evento complete; reproducir nuevo archivo FLV
listenerObject.complete = function(eventObject:Object) {
  if (my_FLVPlybk.contentPath != "http://www.helpexamples.com/flash/video/
  water.flv") {
    my_FLVPlybk.play("http://www.helpexamples.com/flash/video/water.flv");
  }
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.closeVideoPlayer(), FLVPlayback.contentPath, FLVPlayback.load(), FLVPlayback.play(), FLVPlayback.visibleVideoPlayerIndex,

FLVPlayback.closeVideoPlayer()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVPlybk.closeVideoPlayer(*index*:Number)

Parámetros

index Número que representa el índice del reproductor de vídeo que se va a cerrar.

Valor devuelto

El objeto VideoPlayer que se ha cerrado.

Descripción

Método; cierra NetStream y elimina el reproductor de vídeo especificado por el parámetro *index*. Si el reproductor de vídeo cerrado es el reproductor de vídeo activo o visible, la instancia de FLVPlayback establece el reproductor predeterminado (con índice 0) como reproductor de vídeo activo o visible. No se puede cerrar el reproductor predeterminado; si lo intenta, el componente emitirá un error.

Ejemplo

El siguiente ejemplo crea dos reproductores de vídeo para reproducir dos archivos FLV de forma consecutiva en una sola instancia de FLVPlayback. Cuando finaliza el segundo archivo FLV, el controlador de eventos del evento `complete` llama al método `closeVideoPlayer()` para cerrar el segundo reproductor. Si hace clic en el botón Play para reproducir los archivos FLV por segunda vez, verá que el segundo reproductor de vídeo ha desaparecido, y el componente emitirá un error (VideoError) y mostrará un mensaje que indica que la instancia de FLVPlayback no encuentra el archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
// especificar nombre y ubicación del archivo FLV correspondiente al
  reproductor predeterminado
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // añadir un segundo reproductor de vídeo y especificar el nombre y la
  ubicación de su archivo FLV
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // restablecer el reproductor de vídeo predeterminado, que reproduce su
  archivo FLV automáticamente
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
  // si complete hace referencia al segundo archivo FLV, convertir al
  predeterminado en activo y visible
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
    my_FLVPlybk.closeVideoPlayer(1);// cerrar segundo reproductor de vídeo
  } else { // convertir al segundo reproductor en activo y visible y
  reproducir archivo FLV
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// añadir detector de evento complete
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Véase también

FLVPlayback.close, FLVPlayback.activeVideoPlayerIndex, FLVPlayback.visibleVideoPlayerIndex

FLVPlayback.complete

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("complete", listenerObject);
```

Descripción

Evento; se distribuye cuando finaliza la reproducción porque el reproductor alcanzó el final del archivo FLV. El componente no distribuye el evento si se llama a los métodos `stop()` o `pause()` o si se hace clic en los controles correspondientes. El objeto de evento tiene las propiedades `state` y `playheadTime`.

Si la aplicación utiliza descarga progresiva, no establece explícitamente el valor de la propiedad `totalTime` y descarga un archivo FLV que no especifica la duración en los metadatos, y el reproductor de vídeo establece `totalTime` en un valor total aproximado antes de que distribuya este evento.

El reproductor de vídeo también distribuye los eventos `stateChange` y `stopped`.

Ejemplo

En el ejemplo siguiente se utiliza la propiedad `playheadTime` para mostrar en el panel Salida el tiempo de reproducción del archivo FLV transcurrido cuando se produce el evento `complete`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  trace("Elapsed play time at completion is: " + my_FLVPlybk.playheadTime);
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stopped`, `FLVPlayback.totalTime`

FLVPlayback.CONNECTION_ERROR

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`mx.video.FLVPlayback.CONNECTION_ERROR`

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"connectionError"`. Puede comparar esta propiedad con la propiedad `state` para determinar si se ha producido un estado de error de conexión.

Ejemplo

En el siguiente ejemplo se desencadena un error de conexión al especificar un nombre de archivo FLV no válido (`nosuch.flv`) en la propiedad `contentPath`. En el ejemplo se utiliza la propiedad `CONNECTION_ERROR` para detectar el error en un detector del evento `stateChange`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  no_such.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.state == FLVPlayback.CONNECTION_ERROR)
    trace("State: " + FLVPlayback.CONNECTION_ERROR);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.contentPath

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.contentPath`

Descripción

Propiedad; cadena que especifica la URL del archivo FLV que se va a transmitir y cómo se debe transmitir. La URL puede ser una URL HTTP de un archivo FLV, una URL RTMP de un flujo o una URL HTTP de un archivo XML.

Si establece esta propiedad a través del inspector de componentes o del inspector de propiedades, el archivo FLV inicia su carga y reproducción en el siguiente evento `MovieClip.onEnterFrame`. El retardo proporciona tiempo para establecer los valores de las propiedades `isLive`, `autoPlay` y `cuePoints`, entre otras, mientras se realiza la carga. También permite manipular el componente FLVPlayback mediante código ActionScript colocado en el primer fotograma antes de que se inicie la reproducción.

Si establece esta propiedad mediante código ActionScript, la instancia de FLVPlayback cierra el archivo FLV actual e inicia inmediatamente la carga del nuevo archivo FLV. Las propiedades `autoPlay`, `totalTime` y `isLive` afectan al modo en que se carga el nuevo archivo FLV, de modo que, si establece estas propiedades, debe hacerlo *antes* de establecer la propiedad contentPath.

Establezca la propiedad `autoPlay` en `false` para evitar que el nuevo archivo FLV se reproduzca automáticamente.

Ejemplo

En el ejemplo siguiente se establece el valor de la propiedad `contentPath` en ActionScript para especificar la ubicación del archivo FLV que se va a reproducir.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. *No* asigne valores al parámetro `contentPath` del inspector de componentes. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
  my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
}
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
```

Véase también

FLVPlayback.autoPlay, FLVPlayback.isLive, FLVPlayback.play(), FLVPlayback.totalTime

FLVPlayback.cuePoint

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("cuePoint", listenerObject);
```

Descripción

Evento; se distribuye cuando se llega a un cuepoint. El objeto de evento tiene una propiedad `info` que contiene el objeto `info` recibido mediante la función callback `NetStream.onCuePoint` para los cuepoints del archivo FLV. En los cuepoints de ActionScript, contiene el objeto que se pasó en los métodos o propiedades de los objetos cuepoint de ActionScript.

Este evento tiene una propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento.

Ejemplo

En el ejemplo siguiente se añaden dos objetos cuepoint de ActionScript a un archivo FLV. El ejemplo añade el primero mediante un parámetro *`cuePoint`* y el segundo mediante los parámetros *`time`* y *`name`*. Cuando se produce cada uno de los cuepoints, un detector de eventos `cuePoint` muestra el valor de la propiedad `playheadTime` en un área de texto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_ta.visible = false;
// crear objeto cuepoint
var cuePt:Object = new Object(); // crear objeto cuepoint
cuePt.time = 1.444;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
// añadir segundo objeto cuepoint de AS mediante los parámetros time y name
my_FLVPlybk.addASCuePoint(5.3, "elapsed_time2");
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_ta.text = "Cue at: " + eventObject.info.time + " occurred";
    my_ta.visible = true;
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.cuePoints

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.cuePoints`

Descripción

Propiedad; matriz que describe objetos cuepoint de ActionScript y cuepoints de archivos FLV incorporados desactivados. Esta propiedad se creó específicamente para usarse en el inspector de componentes y no funcionará si se establece de otra manera. Su valor sólo produce efecto en el primer archivo FLV cargado, y sólo si se carga estableciendo la propiedad `contentPath` en el inspector de componentes o en el inspector de propiedades.

NOTA	No es posible acceder a esta propiedad mediante ActionScript. Para acceder a la información de cuepoint en ActionScript, utilice la propiedad `metadata`.

Para añadir, quitar, activar o desactivar cuepoints con ActionScript, utilice `addASCuePoint()`, `removeASCuePoint()` o `setFLVCuePointEnabled()`.

Véase también

`FLVPlayback.contentPath`, `FLVPlayback.addASCuePoint()`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.metadata`, `FLVPlayback.metadataReceived`, `FLVPlayback.removeASCuePoint()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.DISCONNECTED

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.DISCONNECTED
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"disconnected"`. Puede comparar esta propiedad con la propiedad `state` para determinar si se produce un estado de desconexión.

La instancia de FLVPlayback está en estado de desconexión hasta que se establece la propiedad `contentPath`.

Ejemplo

El siguiente ejemplo muestra simplemente un mensaje en el panel Salida que confirma que la instancia de FLVPlayback tiene un estado de desconexión antes de establecer la propiedad `contentPath`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;

if(my_FLVPlybk.state == FLVPlayback.DISCONNECTED)
  trace("FLVPlayback instance is currently disconnected");
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.contentPath`, `FLVPlayback.state`

FLVPlayback.EVENT

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.EVENT
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"event"`. Puede utilizar esta propiedad como parámetro `type` para los métodos `findCuePoint()` y `findNearesstCuePoint()`.

Ejemplo

El siguiente ejemplo utiliza la propiedad `FLVPlayback.EVENT` para especificar que desea buscar un cuepoint denominado `myCue`, de tipo `event`. Muestra las propiedades `name`, `time` y `type` del cuepoint devuelto, y el detector de `cuePoint` muestra el mensaje "Hit it!" en el panel Salida cuando se produce el cuepoint.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  if(rtn_cuePt = my_FLVPlybk.findCuePoint("myCue", FLVPlayback.EVENT)){
    trace("Cue point name is: " + rtn_cuePt.name);
    trace("Cue point time is: " + rtn_cuePt.time);
    trace("Cue point type is: " + rtn_cuePt.type);
  }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
  if(eventObject.info.name == "myCue")
    trace("Hit it!");
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

`FLVPlayback.findCuePoint()`

FLVPlayback.fastForward

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.fastForward = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("fastForward", listenerObject);
```

Evento; se distribuye cuando la ubicación de la cabeza lectora se mueve hacia delante al realizar búsquedas, manualmente o mediante código ActionScript, o al hacer clic en el control ForwardButton. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`. La propiedad `playheadTime` indica el tiempo de destino y la propiedad `vp` es el número de índice del reproductor de vídeo al que se aplica el evento.

La instancia de FLVPlayback también distribuye los eventos `seek` y `playheadUpdate`.

Ejemplo

En el ejemplo siguiente se capturan instancias del evento `fastForward` mientras se produce y se muestra el tiempo transcurrido de la cabeza lectora en el panel Salida. Cuando se produce el evento `ready`, una llamada al método `seekPercent()` activa el evento `fastForward`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekPercent(35);
};
my_FLVPlybk.addEventListener("ready", listenerObject);

listenerObject.fastForward = function(eventObject:Object):Void {
  trace("fastforward event; playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("fastForward", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.forwardButton, FLVPlayback.seek, FLVPlayback.seek(), FLVPlayback.seekBar, FLVPlayback.seekPercent(), FLVPlayback.seekSeconds(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.seekToPrevOffset, FLVPlayback.state, FLVPlayback.playheadTime

FLVPlayback.findCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.findCuePoint(time:Number[, type:String]):Object
my_FLVplybk.findCuePoint(name:String[, type:String]):Object
my_FLVplybk.findCuePoint(cuePoint:Object[, type:String]):Object
```

Parámetros

time Número que representa el tiempo (en segundos) del cuepoint objeto de la búsqueda. El método sólo utiliza los primeros tres decimales y redondea los decimales adicionales. El método devuelve el cuepoint correspondiente a dicho tiempo. Si hay varios objetos cuepoint de ActionScript con el mismo tiempo, el método sólo devuelve el primer nombre por orden alfabético. Devuelve `null` si no encuentra ninguna correspondencia.

name Cadena que indica el nombre del cuepoint objeto de la búsqueda. El método devuelve el primer cuepoint correspondiente a dicho nombre o devuelve `null` si no encuentra ninguna correspondencia.

cuePoint Objeto cuepoint que contiene las propiedades `time` y `name` objeto de la búsqueda. Si la propiedad `name` no tiene ningún valor o si éste es `null`, la búsqueda se comporta como si el parámetro fuera un número que representa el tiempo objeto de la búsqueda. Si la propiedad `time` no tiene ningún valor o si éste es `null` o menor que cero, la búsqueda se comporta como si el parámetro fuera una cadena que contiene el nombre objeto de la búsqueda. Si proporciona las propiedades `time` y `name` y hay un objeto cuepoint correspondiente a estos valores, el método lo devuelve. De lo contrario, el método devuelve `null`.

type Opcional. Cadena que especifica el tipo de cuepoint objeto de la búsqueda. Los valores posibles para esta parámetro son: `"actionscript"`, `"all"`, `"event"`, `"flv"` o `"navigation"`. Puede especificar estos valores mediante las siguientes propiedades de clase: `FLVPlayback.ACTIONSCRIPT`, `FLVPlayback.ALL`, `FLVPlayback.EVENT`, `FLVPlayback.FLV` y `FLVPlayback.NAVIGATION`. Si no se especifica este parámetro, el valor predeterminado es `"all"`, lo que significa que el método buscará todos los tipos de cuepoint.

Valor devuelto

Objeto que es una copia del objeto cuepoint encontrado, con las siguientes propiedades adicionales:

`array` Matriz de cuepoints en los que se buscó. Debe tratar esta matriz como de sólo lectura, ya que si añade, elimina o edita los objetos que contiene podría provocar un funcionamiento incorrecto de los cuepoints.

`index` Índice en la matriz del cuepoint devuelto.

Devuelve `null` si no encuentra ninguna correspondencia.

Descripción

Método; busca el cuepoint que tiene el tipo especificado en el parámetro *type* y los parámetros time y name, o la combinación de ambos, especificados.

Si no proporciona un valor para el parámetro time o name del cuepoint, o si el valor del parámetro time es `null`, undefined o es menor que cero, *y* el valor de name es `null` o undefined, el método emite el error 1002 de VideoError. Para más información, consulte "Clase VideoError" en la página 722.

El método incluye en la búsqueda los cuepoints desactivados. Utilice el método `isFLVCuePointEnabled()` para determinar si un cuepoint está desactivado.

Ejemplo

El siguiente ejemplo añade dos objetos cuepoint de ActionScript a un archivo FLV y llama tres veces al método `findCuePoint()`. La primera llamada busca un cuepoint de navegación cuyo valor de tiempo es 7.748. La segunda llamada busca el cuepoint `"ASCue1"` únicamente a través de un valor de nombre. La tercera llamada utiliza un objeto cuepoint que especifica tanto un tiempo, `10`, como un nombre, `"ASCue2"`. Después de la tercera llamada, el ejemplo muestra el contenido del objeto de cuepoint devuelto, incluida la matriz de cuepoints objeto de la búsqueda que, en este ejemplo, fueron objetos cuepoint de ActionScript.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
// crear objeto cuepoint
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object();
var rtn_cuePt:Object = new Object(); // crear objeto para valor devuelto
cuePt.time = 4.444;
cuePt.name = "AScue1";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
// añadir segundo objeto cuepoint de AS mediante los parámetros time y name
my_FLVPlybk.addASCuePoint(10, "AScue2");
// buscar cuepoint de navegación mediante el parámetro time
rtn_cuePt = my_FLVPlybk.findCuePoint(7.748, FLVPlayback.NAVIGATION);
// buscar cuepoint de ActionScript sólo mediante el parámetro name
rtn_cuePt = my_FLVPlybk.findCuePoint("AScue1");
// buscar cuepoint de ActionScript mediante el objeto cuepoint
cuePt.time = 10;
cuePt.name = "AScue2";
rtn_cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.ACTIONSCRIPT);
// ver contenido del objeto cuepoint devuelto
for (i in rtn_cuePt) {
  //si es un objeto de matriz, abrirlo y localizar el contenido
  if (typeof rtn_cuePt[i] == "object") {
    tracer(rtn_cuePt[i]);
  // si no, localizar par de nombre (i) y valor (rtn_cuePt[i])
  } else {
    trace(i+ " " + rtn_cuePt[i]);
  }
}
// localizar matriz de cuepoints
function tracer(cuepts:Array) {
  for (i in cuepts) {
    if (typeof cuepts[i] == "object") { // si el objeto es object
        tracer(cuepts[i]); //localizar objeto
        } else {
       // localizar el par nombre : valor
        trace(i + " " + cuepts[i]);
        }
  }
}
```

Véase también

FLVPlayback.addASCuePoint(), FLVPlayback.cuePoints, FLVPlayback.findNearestCuePoint(), FLVPlayback.findNextCuePointWithName(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.removeASCuePoint(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.findNearestCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.findNearestCuePoint(time:Number[, type:String]):Object
my_FLVplybk.findNearestCuePoint(name:String[, type:String]):Object
my_FLVplybk.findNearestCuePoint(cuePoint:Object[, type:String]):Object
```

Parámetros

time Número que representa el tiempo (en segundos) del cuepoint objeto de la búsqueda. El método sólo utiliza los primeros tres decimales y redondea los decimales adicionales. El método devuelve el cuepoint correspondiente a este tiempo o el cuepoint *anterior* más cercano del tipo especificado. Si hay varios objetos cuepoint con el mismo tiempo, lo cual sólo es posible en los objetos cuepoint de ActionScript, el método sólo devuelve el primer nombre por orden alfabético. Devuelve `null` si no encuentra ninguna correspondencia.

name Cadena que indica el nombre del cuepoint objeto de la búsqueda. El método devuelve el primer cuepoint correspondiente a dicho nombre o devuelve `null` si no encuentra ninguna correspondencia.

cuePoint Objeto cuepoint que contiene las propiedades `time` y `name` objeto de la búsqueda. Si se proporciona un tiempo y la propiedad `name` no tiene ningún valor o si éste es `null`, la búsqueda se comporta como si el parámetro fuera un número que representa el tiempo objeto de la búsqueda. Si se proporciona un nombre y la propiedad `time` no tiene ningún valor o si éste es `null` o menor que cero, la búsqueda se comporta como si el parámetro fuera una cadena que contiene el nombre objeto de la búsqueda. Si proporciona las propiedades `time` y `name` y hay un objeto cuepoint correspondiente a estos valores, el método lo devuelve. Si no encuentra una correspondencia de tiempo y nombre, devuelve el primer cuepoint que coincide con el nombre y que tiene un tiempo *anterior*. Si no hay ningún cuepoint anterior con ese nombre, devuelve el primero que coincide con ese nombre. De lo contrario, el método devuelve `null`.

type Opcional. Cadena que especifica el tipo de cuepoint objeto de la búsqueda. Los valores posibles para esta parámetro son: `"actionscript"`, `"all"`, `"event"`, `"flv"` o `"navigation"`. Puede especificar estos valores mediante las siguientes propiedades de clase: `FLVPlayback.ACTIONSCRIPT`, `FLVPlayback.ALL`, `FLVPlayback.EVENT`, `FLVPlayback.FLV` y `FLVPlayback.NAVIGATION`. Si no se especifica este parámetro, el valor predeterminado es `"all"`, lo que significa que el método buscará todos los tipos de cuepoint.

Valor devuelto

Objeto que es una copia del objeto cuepoint encontrado, con las siguientes propiedades adicionales:

`array` Matriz de cuepoints objeto de la búsqueda. Debe tratar esta matriz como de sólo lectura, ya que si añade, elimina o edita los objetos que contiene podría provocar un funcionamiento incorrecto de los cuepoints.

`index` Índice en la matriz del cuepoint devuelto.

Devuelve `null` si no encuentra ninguna correspondencia.

Descripción

Método; busca un cuepoint del tipo especificado, correspondiente al tiempo especificado (*o anterior* a él) o que coincida con el nombre especificado, si especifica ambos parámetros y no hay ningún cuepoint anterior que coincida con ese nombre. De lo contrario, devuelve `null`.

El método incluye en la búsqueda los cuepoints desactivados. Utilice el método `isFLVCuePointEnabled()` para determinar si un cuepoint está desactivado.

Si el valor de time es `null`, undefined o es menor que 0, *y* el valor de name es `null` o undefined, el método emite un error VideoError (1002).

Ejemplo

El siguiente ejemplo crea un objeto cuepoint de ActionScript para el archivo FLV en el tiempo correspondiente a 4,07 segundos. Cuando se produce este cuepoint, el controlador de eventos `cuePoint` llama al método `findNearestCuePoint()` para buscar un cuepoint de cualquier tipo que se encuentre más cercano a un tiempo correspondiente a 5 segundos después. El panel Salida muestra el nombre, el tiempo y el tipo del objeto cuepoint devuelto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object(); // crear objeto cuepoint
cuePt.time = 4.07;
cuePt.name = "ASpt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
function cuePoint(eventObject:Object):Void {
  if (eventObject.info.name == "ASpt1") {
    var rtn_obj:Object = new Object();
    rtn_obj = my_FLVPlybk.findNearestCuePoint(eventObject.info.time + 5);
    trace("Cue point name is: " + rtn_obj.name);
    trace("Cue point time is: " + rtn_obj.time);
    trace("Cue point type is: " + rtn_obj.type);
  }
}
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
```

Véase también

`FLVPlayback.addASCuePoint()`, `FLVPlayback.cuePoints`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.removeASCuePoint()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.findNextCuePointWithName()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.findNextCuePointWithName(*my_cuePoint*:Object)

Parámetros

my_cuePoint Objeto cuepoint devuelto por el método `findCuePoint()`, el método `findNearestCuePoint()` o una llamada anterior a este método.

Valor devuelto

Objeto que es una copia del objeto cuepoint encontrado, con las siguientes propiedades adicionales:

`array` Matriz de cuepoints objeto de la búsqueda. Debe tratar esta matriz como de sólo lectura, ya que si añade, elimina o edita los objetos que contiene podría provocar un funcionamiento incorrecto de los cuepoints.

`index` Índice en la matriz del cuepoint devuelto.

Devuelve `null` si no encuentra ninguna correspondencia.

Descripción

Método; busca el siguiente cuepoint en *my_cuePoint*.`array` que tenga el mismo nombre que *my_cuePoint*.`name`. El objeto *my_cuePoint* debe ser un objeto cuepoint devuelto por el método `findCuePoint()`, el método `findNearestCuePoint()` o una llamada anterior a este método. Este método utiliza la propiedad `array` que estos métodos añaden al objeto cuepoint.

El método incluye en la búsqueda los cuepoints desactivados. Utilice el método `isFLVCuePointEnabled()` para determinar si un cuepoint está desactivado.

Devuelve `null` si no hay más objetos cuepoint en la matriz con un nombre coincidente.

Ejemplo

El siguiente ejemplo crea tres objetos cuepoint de ActionScript con el nombre `"transition"`. Cuando se produce el evento `ready`, el controlador de eventos llama al método `findCuePoint()` para buscar el primer cuepoint con ese nombre. Si encuentra una correspondencia, llama a la función `findNextName()`, que llama al método `findNextCuePointWithName()` y pasa el objeto cuepoint devuelto (`rtn_obj`), para buscar cuepoints adicionales con el mismo nombre.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object(); // crear objeto cuepoint
cuePt.time = 6.27;
cuePt.name = "transition";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
cuePt.time = 7.06;
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
cuePt.time = 11.13;
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
var listenerObject:Object = new Object();
listenerObject.ready = function():Void {
  var rtn_obj:Object = new Object();
// Buscar cuepoint de mediante la cadena name
  if (rtn_obj = my_FLVPlybk.findCuePoint("transition")) {
    trace("Cue point name is: " + rtn_obj.name);
    trace("Cue point time is: " + rtn_obj.time);
    trace("Cue point type is: " + rtn_obj.type);
    findNextName(rtn_obj);
  }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
// Buscar cuepoints adicionales con el mismo nombre
function findNextName(cuePt:Object):Void {
  while(cuePt = my_FLVPlybk.findNextCuePointWithName(cuePt)) {
       trace("Cue point name is: " + cuePt.name);
       trace("Cue point time is: " + cuePt.time);
       trace("Cue point type is: " + cuePt.type);
  }
}
```

Véase también

FLVPlayback.addASCuePoint(), FLVPlayback.cuePoints, FLVPlayback.findCuePoint(), FLVPlayback.findNearestCuePoint(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.removeASCuePoint(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.FLV

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.FLV
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"flv"`. Puede utilizar esta propiedad como parámetro `type` para los métodos `findCuePoint()` y `findNearestCuePoint()`.

Ejemplo

El siguiente ejemplo busca entre los cuepoints de archivos FLV un cuepoint denominado `point2`, que tenga un tiempo de 7,748, y muestra el tipo y el tiempo encontrados. Los cuepoints de archivos FLV son los cuepoints de navegación y evento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
// crear objeto cuepoint
var listenerObject = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var cuePt:Object = new Object(); // crear objeto cuepoint
  cuePt.name = "point3";
  cuePt.time = 16.02;
  if(cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.FLV)) //buscar
  objeto cuepoint
    trace("found a " + cuePt.type + " cue point at " + cuePt.time);
  else
    trace("cue point not found");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`

FLVPlayback.forwardButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.forwardButton

Descripción

Propiedad; objeto MovieClip que es el control del botón para avanzar. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton` y `stopButton` para asociar controles individuales de interfaz de usuario personalizados de reproducción FLV a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes personalizados de interfaz de usuario de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) y StopButton (**my_stopbttn**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayButton, PauseButton y
   StopButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.fastForward`, `FLVPlayback.seek`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.getVideoPlayer()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`my_FLVplybk``.getVideoPlayer(`*`index`*`:Number)`

Valor devuelto

Un objeto VideoPlayer.

Descripción

Método; obtiene el reproductor de vídeo especificado por su valor de *index*. Si es posible, es mejor acceder a los métodos y las propiedades de VideoPlayer mediante los métodos y las propiedades de FLVPlayback. Cada propiedad `VideoPlayer._name` es su índice.

Ejemplo

El siguiente ejemplo utiliza dos reproductores de vídeo para reproducir dos archivos FLV. Cuando el segundo archivo FLV activa el evento `ready`, el ejemplo llama al método `getVideoPlayer()` para obtener el reproductor de vídeo número 1 y establecer su propiedad `_alpha` en 50. Como consecuencia, el archivo FLV (plane_cuepoints) de dicho reproductor será transparente y permitirá ver al mismo tiempo ambos archivos FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/cuepoints.flv");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
    if (eventObject.target.contentPath == "http://www.helpexamples.com/
  flash/video/cuepoints.flv") {
        //se activa cuando está listo el primer archivo FLV
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.load("http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv");
    } else {
        //se activa cuando está listo el segundo archivo FLV
        eventObject.target.activeVideoPlayerIndex = 0;
        eventObject.target.play();
        eventObject.target.activeVideoPlayerIndex = 1;
        eventObject.target.play();
        var layerOnTop:MovieClip = eventObject.target.getVideoPlayer(1);
        layerOnTop._alpha = 50;
        layerOnTop._visible = true;
    }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.bringVideoPlayerToFront()`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.height

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.height

Descripción

Propiedad; número que especifica la altura de la instancia de FLVPlayback. Esta propiedad sólo afecta a la altura de la instancia de FLVPlayback y no incluye la altura de un archivo SWF de aspecto que pueda cargarse. Utilice la propiedad `height` de FLVPlayback y no la propiedad `MovieClip._height` porque la propiedad `_height` podría devolver un valor distinto si se carga un archivo SWF de aspecto.

Ejemplo

El siguiente ejemplo establece las propiedades `width` y `height` para cambiar el tamaño del reproductor de vídeo. Primero establece la propiedad `maintainAspectRatio` en `false` para evitar que el reproductor de vídeo cambie de tamaño automáticamente si se modifican las dimensiones.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.maintainAspectRatio = false;
my_FLVPlybk.width = 300;
my_FLVPlybk.height = 350;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.maintainAspectRatio`, `FLVPlayback.resize`, `FLVPlayback.setSize()`, `FLVPlayback.width`

FLVPlayback.idleTimeout

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.idleTimeout

Descripción

Propiedad; cantidad de tiempo en milisegundos antes de que Flash cierre una conexión inactiva al servidor FCS tras pausar o detener una reproducción. Esta propiedad no produce ningún efecto en la descarga de un archivo FLV a través de HTTP.

Si se establece esta propiedad cuando un flujo de vídeo ya está inactivo, se reinicia el período de tiempo de espera con el nuevo valor.

El valor predeterminado es 300.000 (5 minutos).

Ejemplo

En el siguiente ejemplo se supone que se reproduce un archivo FLV transmitido desde un servidor FCS o FVSS. El ejemplo establece la propiedad `idleTimeout` en un valor bajo (10 milisegundos), lo que provoca que se agote el tiempo de espera y, en consecuencia, activa un evento `close` en la conexión RTMP. El detector del evento `close` muestra el número de índice del reproductor de vídeo afectado por el evento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. En el inspector de componentes, asigne al parámetro `contentPath` un valor que especifique la ubicación de un archivo FLV transmitido desde un servidor FCS o FVSS. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.idleTimeout = 10;
var listenerObject:Object = new Object();
// detectar evento close en conexión RTMP; mostrar índice del reproductor de
  vídeo
listenerObject.close = function(eventObject:Object) {
  trace("Closed connection for video player: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("close", listenerObject);
```

Véase también

`FLVPlayback.close`

FLVPlayback.isFLVCuePointEnabled()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.isFLVCuePointEnabled(time:Number)
my_FLVplybk.isFLVCuePointEnabled(name:String)
my_FLVplybk.isFLVCuePointEnabled(cuePoint:Object)
```

Parámetros

time Número que representa el tiempo (en segundos) del cuepoint objeto de la búsqueda.

name Cadena que indica el nombre del cuepoint objeto de la búsqueda.

cuePoint Objeto cuepoint con propiedades `time` y `name` correspondientes al cuepoint. El método no comprueba ninguna otra propiedad en el objeto cuepoint entrante. Si el valor de `time` o `name` no está definido, el método sólo utiliza la propiedad que está definida.

Valor devuelto

Un valor booleano. Es `false` si se encuentran los cuepoints y están desactivados, y `true` si no están desactivados o no existen. Si el valor del parámetro time es undefined, `null`, menor que 0, o sólo se proporciona un nombre de cuepoint, el método devuelve `false` sólo si todos los cuepoints con este nombre están desactivados.

Descripción

Método; devuelve `false` si el cuepoint incorporado del archivo FLV está desactivado. Puede desactivar cuepoints estableciendo la propiedad `cuePoints` mediante el cuadro de diálogo Cuepoints de Flash Video o mediante una llamada al método `setFLVCuePointEnabled()`.

El valor devuelto por esta función sólo es significativo cuando el valor de la propiedad `metadataLoaded` es `true`, el valor de la propiedad `metadata` no es `null`, o después de un evento `metadataReceived`. Si el valor de `metadataLoaded` es `false`, esta función siempre devuelve `true`.

Ejemplo

El siguiente ejemplo desactiva el cuepoint `point2` cuando se produce un evento `ready`. Sin embargo, cuando se produce el primer evento `cuePoint`, el controlador de eventos llama al método `isFLVCuePointEnabled()` para ver si el cuepoint está desactivado y, si lo está, lo activa. El archivo FLV contiene los siguientes cuepoints incorporados: point1, 00:00:00:418; point2, 00:00:07.748; point3, 00:00:16:020.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
function ready(eventObject:Object) {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
}
my_FLVPlybk.addEventListener("ready", ready);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    trace("Elapsed time in seconds: " + my_FLVPlybk.playheadTime);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
    if (my_FLVPlybk.isFLVCuePointEnabled("point2") == false) {
      my_FLVPlybk.setFLVCuePointEnabled(true, "point2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

FLVPlayback.cuePoint, FLVPlayback.findCuePoint(), FLVPlayback.findNearestCuePoint(), FLVPlayback.findNextCuePointWithName(), FLVPlayback.setFLVCuePointEnabled(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint()

FLVPlayback.isLive

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.isLive

Descripción

Propiedad; valor booleano. Es `true` si el flujo de vídeo es dinámico. Esta propiedad sólo es efectiva cuando se transmite desde un servidor FCS o FVSS. El valor de esta propiedad se omitirá para una descarga HTTP.

Si establece esta propiedad mientras se cargan nuevos archivos FLV, no tendrá ningún efecto hasta que se establezca el valor del parámetro `contentPath` para el nuevo archivo FLV.

Ejemplo

En el siguiente ejemplo se supone que se reproduce un flujo en vivo desde un servidor FCS. Cuando se produce el evento `playing`, el ejemplo muestra el valor de la propiedad `isLive`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo FLV en el servidor FCS.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playing = function(eventObject:Object) {
  trace("The isLive property is " + my_FLVPlybk.isLive);
};
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Véase también

FLVPlayback.contentPath, FLVPlayback.load()

FLVPlayback.isRTMP

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.isRTMP

Descripción

Propiedad; valor booleano. Es `true` si el archivo FLV se transmite desde un servidor FCS o FVSS. Su valor es `false` en cualquier otro origen de archivo FLV. Sólo lectura.

Ejemplo

En el siguiente ejemplo se supone que se reproduce un archivo FLV transmitido desde un servidor FCS o FVSS. Cuando se produce el evento `playing`, el ejemplo muestra el valor de la propiedad `isRTMP` para indicar si el archivo FLV procede de una URL RTMP.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo. En la sentencia que carga la propiedad `contentPath`, reemplace el texto en cursiva por el nombre y la ubicación del archivo FLV en el servidor FCS.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 7;
var listenerObject:Object = new Object();
// detectar evento playing en conexión RTMP; mostrar resultado de isRTMP
listenerObject.playing = function(eventObject:Object) {
  trace("Value of isRTMP property is: " + my_FLVPlybk.isRTMP);
};
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Véase también

FLVPlayback.contentPath, FLVPlayback.load(), FLVPlayback.play()

FLVPlayback.load()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.load(*contentPath*:String[, *totalTime*:Number, *isLive*:Boolean])

Parámetros

contentPath Cadena que especifica la URL del archivo FLV que se va a transmitir y cómo se debe transmitir. La URL puede ser una URL HTTP de un archivo FLV, una URL RTMP de un flujo de archivo FLV o una URL HTTP de un archivo XML.

totalTime Número que representa el tiempo total de reproducción del vídeo. Opcional.

isLive Valor booleano. Es `true` si el flujo de vídeo es dinámico. Este valor sólo es efectivo cuando se transmite desde un servicio FVSS o un servidor FCS. El valor de esta propiedad se omitirá para una descarga HTTP. Opcional

Valor devuelto

Ninguno.

Descripción

Método; inicia la carga del archivo FLV y proporciona un acceso directo para establecer el valor de la propiedad `autoPlay` en `false` y para establecer los valores de las propiedades `contentPath`, `totalTime` y `isLive`, si se proporcionan. Si la propiedad `totalTime` o la propiedad `isLive` no están definidas, no se establecerán. Si el valor de `contentPath` es undefined, `null` o es una cadena vacía, este método no hace nada.

Ejemplo

El ejemplo siguiente llama al método `load()` para cargar un archivo FLV especificado por el parámetro `contentPath`. Muestra el valor de de la propiedad `autoPlay` antes y después de cargar el archivo FLV y llama al método `play()` para iniciar la reproducción del archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
trace("Before load, autoPlay is: " + my_FLVPlybk.autoPlay);
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/water.flv");
trace("After load, autoPlay is: " + my_FLVPlybk.autoPlay);
my_FLVPlybk.play();
```

Véase también

`FLVPlayback.contentPath`, `FLVPlayback.isLive`, `FLVPlayback.totalTime`

FLVPlayback.LOADING

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.LOADING
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"loading"`. Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado de carga.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `FLVPlayback.LOADING` si el archivo FLV está en estado de carga.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.state == FLVPlayback.LOADING)
    trace("State is " + FLVPlayback.LOADING);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.maintainAspectRatio

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.maintainAspectRatio`

Descripción

Propiedad; valor booleano. Si es `true`, mantiene la proporción de vídeo. Si se cambia el valor de esta propiedad de `false` a `true` y el valor de la propiedad `autoSize` es `false` después de cargar un archivo FLV, se iniciará inmediatamente un cambio de tamaño automático del vídeo. El valor predeterminado es `true`.

Ejemplo

El siguiente ejemplo llama a `setSize()` para cambiar el tamaño de la instancia de FLVPlayback, lo cual activa un evento `resize`. La propiedad `maintainAspectRatio`, que tiene como valor predeterminado `true`, activa un segundo evento `resize` para mantener la proporción. El controlador de eventos resize muestra en el panel Salida la anchura y la altura de la instancia de FLVPlayback a la que se ha cambiado el tamaño, para ambos eventos. Si se establece `maintainAspectRatio` en `false`, las dimensiones especificadas por el método `setSize()` surten efecto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// maintainAspectRatio tiene true como valor predeterminado, lo cual activa
  un evento resize cuando cambia el tamaño.
// Eliminar delimitadores de comentario de la siguiente línea para
  desactivar evento resize.

// my_FLVPlybk.maintainAspectRatio = false;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  trace("resize event; Width is: " + eventObject.target.width + " Height
  is: " + eventObject.target.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.setSize(300, 300);
```

Véase también

`FLVPlayback.autoSize`, `FLVPlayback.height`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.resize`, `FLVPlayback.setSize()`, `FLVPlayback.width`

FLVPlayback.metadata

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.metadata

Descripción

Propiedad; objeto que es un paquete de información de metadatos que se recibe de una llamada a la función callback `NetSteam.onMetaData()`, si está disponible. Sólo lectura.

Si el archivo FLV se codifica con el codificador de Flash 8, la propiedad `metadata` contiene la siguiente información. Los archivos FLV antiguos sólo contienen los valores de `height`, `width` y `duration`.

Parámetro	Descripción
`canSeekToEnd`	Valor booleano. Es `true` si el archivo FLV se codifica con un fotograma clave en el último fotograma, lo que permite buscar hasta el final de un clip de película de descarga progresiva. Es `false` si el archivo FLV no se codifica con un fotograma clave en el último fotograma.
`cuePoints`	Matriz de objetos, uno por cada cuepoint incorporado en el archivo FLV. El valor es undefined si el archivo FLV no contiene ningún cuepoint. Cada objeto tiene las siguientes propiedades: • `type` cadena que especifica el tipo de cuepoint como `"navigation"` o `"event"`. • `name` Cadena que indica el nombre del cuepoint. • `time` Número que indica el tiempo del cuepoint en segundos, con una precisión de tres decimales (milisegundos). • `parameters` Objeto opcional que tiene pares nombre-valor designados por el usuario durante la creación de los cuepoints.
`audiocodecid`	Número que indica el códec de audio (técnica de codificación/descodificación) que se ha utilizado.
`audiodelay`	Número que indica el tiempo del archivo FLV correspondiente al tiempo 0 (“time 0”) del archivo FLV original. Es necesario retardar ligeramente el contenido del vídeo para sincronizarlo correctamente con el audio.
`audiodatarate`	Número que indica los kilobytes por segundo de audio.

Parámetro	Descripción
`videocodecid`	Número que es la versión de códec que se utilizó para codificar el vídeo.
`framerate`	Número que especifica la velocidad de fotogramas del archivo FLV.
`videodatarate`	Número que especifica la velocidad de datos de vídeo del archivo FLV.
`height`	Número que especifica la altura del archivo FLV.
`width`	Número que especifica la anchura del archivo FLV.
`duration`	Número que especifica la duración del archivo FLV en segundos.

Ejemplo

El siguiente ejemplo muestra en el panel Salida distintos valores `metadata` de muestra del archivo FLV cuepoints.flv. Muestra los datos cuando se produce el evento `metadataReceived`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
    trace("canSeekToEnd is " + my_FLVPlybk.metadata.canSeekToEnd);
    trace("Number of cue points is " +
  my_FLVPlybk.metadata.cuePoints.length);
    trace("Frame rate is " + my_FLVPlybk.metadata.framerate);
    trace("Height is " + my_FLVPlybk.metadata.height);
    trace("Width is " + my_FLVPlybk.metadata.width);
    trace("Duration is " + my_FLVPlybk.metadata.duration + " seconds");
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.metadataLoaded`, `FLVPlayback.metadataReceived`

FLVPlayback.metadataLoaded

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.metadataLoaded

Descripción

Propiedad; valor booleano. Es `true` si se ha encontrado y procesado un paquete de metadatos *o* si el archivo FLV se ha codificado sin el paquete de metadatos. Dicho de otro modo, el valor es `true` si se reciben los metadatos o nunca se van a obtener metadatos. Así, puede saber si tiene los metadatos y, si no los tiene, sabe que no debe esperar recibirlos. Si simplemente desea saber si tiene metadatos o no, puede comprobar el valor con:

```
FLVPlayback.metadata != null
```

Utilice esta propiedad para comprobar si puede obtener información útil con los métodos para buscar y activar o desactivar cuepoints. Sólo lectura.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `progress`. Cuando se produce el evento, el ejemplo comprueba si la propiedad `metadataLoaded` es `true` y, si lo es, muestra los valores de metadatos `height`, `width` y `duration` en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    if(my_FLVPlybk.metadataLoaded){
      trace("Height is " + my_FLVPlybk.metadata.height);
      trace("Width is " + my_FLVPlybk.metadata.width);
      trace("Duration is " + my_FLVPlybk.metadata.duration + " seconds");
    }
};
my_FLVPlybk.addEventListener("progress", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.metadata`, `FLVPlayback.metadataReceived`

FLVPlayback.metadataReceived

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("metadataReceived", listenerObject);
```

Descripción

Evento; se distribuye la primera vez que se llega a los metadatos del archivo FLV. El objeto de evento tiene una propiedad `info` que contiene el objeto `info` recibido mediante la función callback `NetStream.onMetaData`.

El objeto de evento tiene además una propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `metadataReceived`. Cuando se produce el evento, el controlador de eventos envía el nombre, el tiempo y el tipo de cada cuepoint que se describe en la propiedad `metadata` del panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
     var i:Number = 0;
     trace("This FLV contains the following cue points:");
     while(i < my_FLVPlybk.metadata.cuePoints.length) {
        trace("\nName: " + my_FLVPlybk.metadata.cuePoints[i].name);
        trace(" Time: " + my_FLVPlybk.metadata.cuePoints[i].time);
        trace(" Type is " + my_FLVPlybk.metadata.cuePoints[i].type);
        ++i;
     }
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.metadata`, `FLVPlayback.metadataLoaded`

FLVPlayback.muteButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.muteButton`

Descripción

Propiedad; objeto MovieClip que es el control del botón de silencio. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Si se hace clic en el control muteButton, se distribuye un evento `volumeUpdate`.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playPauseButton`, `stopButton` y `muteButton` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes personalizados de interfaz de usuario de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton (**my_plypausbttn**), StopButton (**my_stopbttn**) y MuteButton (**my_mutebttn**). Añada las siguientes líneas de código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayPauseButton, StopButton y MuteButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.muteButton = my_mutebttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.skin`, `FLVPlayback.volume`, `FLVPlayback.volumeBar`, `FLVPlayback.volumeUpdate`

FLVPlayback.NAVIGATION

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`mx.video.FLVPlayback.NAVIGATION`

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"navigation"`. Puede utilizar esta propiedad como parámetro `type` para los métodos `findCuePoint()` y `findNearestCuePoint()`.

Ejemplo

En el siguiente ejemplo se utiliza la propiedad `FLVPlayback.NAVIGATION` para especificar el tipo de cuepoint que se va a buscar.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// buscar cuepoint de navegación mediante el parámetro time
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var rtn_cuePt:Object = new Object();
  rtn_cuePt = my_FLVPlybk.findCuePoint(7.748, FLVPlayback.NAVIGATION);
  trace("Found cue point at " + rtn_cuePt.time + " of type " +
  rtn_cuePt.type);
}
my_FLVPlybk.addEventListener("ready", listenerObject)
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`

FLVPlayback.ncMgr

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.ncMgr`

Descripción

Propiedad; objeto INCManager que proporciona acceso a una instancia de la clase que implementa `INCManager`, que es una interfaz de la clase `NCManager`.

Puede utilizar esta propiedad para implementar un objeto INCManager personalizado que requiere una inicialización personalizada. Sólo lectura.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `DEFAULT_TIMEOUT` de NetConnection cuando se produce el evento `ready`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// especificar nombre y ubicación de FLV
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var NC:Object = new Object();
  NC = my_FLVPlybk.ncMgr;
  trace("Net connection timeout is " + NC.DEFAULT_TIMEOUT + "
  milliseconds");
};
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`Clase VideoPlayer`

FLVPlayback.pause()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`my_FLVplybk``.pause()`

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; hace una pausa en la reproducción del flujo de vídeo.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `playheadUpdate`. Cuando se produce este evento, el controlador de eventos comprueba si el tiempo de la cabeza lectora está entre 5 y 5,05 segundos. En caso afirmativo, el controlador de eventos llama al método `pause()` para suspender la reproducción del archivo FLV. El controlador de eventos `paused` le indica que pulse el botón de reproducción para continuar.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.playheadUpdateInterval = 5;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if ((eventObject.playheadTime >= 5) && (eventObject.playheadTime < 5.05))
  {
    my_FLVPlybk.pause();
  }
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
listenerObject.paused = function(eventObject:Object):Void {
  my_ta.text = "Paused; push Play to continue";
  my_ta.visible = true;
};
my_FLVPlybk.addEventListener("paused", listenerObject);
```

Véase también

`FLVPlayback.paused`, `FLVPlayback.play()`, `FLVPlayback.rewind`

FLVPlayback.pauseButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.pauseButton

Descripción

Propiedad; objeto MovieClip que es el control PauseButton. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton` y `stopButton` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes personalizados de interfaz de usuario de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) y StopButton (**my_stopbttn**). Añada las siguientes líneas de código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayButton, PauseButton y
   StopButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.playButton, FLVPlayback.playPauseButton, FLVPlayback.skin

FLVPlayback.PAUSED

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

mx.video.FLVPlayback.PAUSED

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena "paused". Puede comparar esta propiedad con la propiedad state para determinar si el componente está en estado de pausa.

Ejemplo

El siguiente ejemplo utiliza la propiedad FLVPlayback.PAUSED para mostrar el estado del archivo FLV cuando el usuario hace clic en el botón de pausa.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.PAUSED)
    trace("FLV is " + FLVPlayback.PAUSED);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject)
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.paused

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.paused = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("paused", listenerObject);
```

Descripción

Evento; se distribuye cuando el reproductor pasa al estado de pausa. Se produce al llamar al método `pause()` o al hacer clic en el control correspondiente, y también se produce en algunos casos cuando se carga el archivo FLV si el valor de `autoPlay` es `false` (en su lugar, el estado puede ser `stopped`). El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

También se distribuye el evento `stateChange`.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `playheadUpdate`. Cuando se produce el evento, el controlador de eventos comprueba si el valor de la propiedad `playheadTime` está entre 5 y 5,05 segundos. En caso afirmativo, el controlador de eventos llama al método `pause()` para suspender la reproducción del archivo FLV. Esto activa un evento `paused` y el controlador de eventos `paused` correspondiente muestra un mensaje para indicar que el FLV está en pausa.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.playheadUpdateInterval = 5;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if ((eventObject.playheadTime >= 5) && (eventObject.playheadTime < 5.05))
  {
    my_FLVPlybk.pause();
  }
}
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
listenerObject.paused = function(eventObject:Object) {
  trace("FLV is " + my_FLVPlybk.state + "!");
};
my_FLVPlybk.addEventListener("paused", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.pause(), FLVPlayback.paused, FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.paused

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.paused

Descripción

Propiedad; valor booleano. Es `true` si el archivo FLV está en estado de pausa. Sólo lectura.

Ejemplo

En el ejemplo siguiente se crea un detector para el evento `stateChange`. Cuando se produce el evento, comprueba la propiedad `paused` para determinar si el componente está en estado de pausa. Si lo está, muestra un mensaje para indicarlo en el panel Salida. Debe hacer clic en el botón de pausa mientras se reproduce el archivo FLV para que se produzca el estado de pausa.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object) {
  if(my_FLVPlybk.paused)
    trace("FLV is in " + FLVPlayback.PAUSED + " state");
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.paused`, `FLVPlayback.PAUSED`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.play()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.play ([contentPath:String, totalTime:Number, isLive:Boolean])
```

Parámetros

contentPath Cadena que especifica la URL del archivo FLV que se va a transmitir y cómo se debe transmitir. La URL puede ser una URL HTTP de un archivo FLV, una URL RTMP de un flujo de archivo FLV o una URL HTTP de un archivo XML. Es opcional, pero la propiedad `contentPath` debe establecerse a través del inspector de componentes o mediante código ActionScript para que el método surta efecto.

totalTime Número que representa el tiempo total de reproducción del vídeo. Opcional.

isLive Valor booleano. Es `true` si el flujo de vídeo es dinámico. Este valor sólo es efectivo cuando se transmite desde un servidor FCS o FVSS. El valor de esta propiedad se omitirá para una descarga HTTP. Opcional.

Valor devuelto

Ninguno.

Descripción

Método; reproduce el flujo de vídeo. Sin parámetros, el método sólo pasa el FLV del estado de pausa o el estado detenido al estado de reproducción.

Si se utilizan parámetros, el método actúa como una forma rápida de establecer el valor de la propiedad `autoPlay` en `true` y de establecer los valores de las propiedades `isLive`, `totalTime` y `contentPath`. Si la propiedad `totalTime` o la propiedad `isLive` no están definidas, no se establecerán.

Ejemplo

En el ejemplo siguiente se desactiva la reproducción automática del archivo FLV, se llama al método `seekSeconds()` para mover la cabeza lectora 20 segundos en el vídeo, y se llama al método `play()` para iniciar la reproducción del archivo FLV en ese punto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekSeconds(4);
  my_FLVPlybk.play();
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.autoPlay`, `FLVPlayback.contentPath`, `FLVPlayback.load()`, `FLVPlayback.pause()`, `FLVPlayback.stop()`

FLVPlayback.playButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.playButton

Descripción

Propiedad; objeto MovieClip que es el botón de reproducción. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utiliza las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton` y `stopButton` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) y StopButton (**my_stopbttn**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayButton, PauseButton y
   StopButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
```

Véase también

`FLVPlayback.playing`, `FLVPlayback.skin`

FLVPlayback.playheadPercentage

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.playheadPercentage`

Descripción

Propiedad; número que especifica el valor actual de `playheadTime` como un porcentaje de la propiedad `totalTime`. Si accede a esta propiedad, verá que contiene el porcentaje de tiempo de reproducción transcurrido. Si establece esta propiedad, provoca una operación de búsqueda en el punto que representa dicho porcentaje del tiempo de reproducción del archivo FLV.

El valor de esta propiedad es relativa al valor de la propiedad `totalTime`.

El componente emite un error VideoError si especifica un porcentaje no válido o si el valor de la propiedad `totalTime` es undefined, null, o menor o igual que cero.

Ejemplo

El siguiente ejemplo muestra el porcentaje del archivo FLV que se ha reproducido cuando se produce el cuepoint `point2`. En el cuepoint `point3`, establece `playheadPercentage` en `10`, lo que provoca una operación de búsqueda en el punto correspondiente al 10% con respecto al inicio del archivo FLV y la creación de un bucle de reproducción.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    if(eventObject.info.name == "point2")
      trace("point2 occurred at " + my_FLVPlybk.playheadPercentage + "
  percent of FLV");
    if(eventObject.info.name == "point3")
      my_FLVPlybk.playheadPercentage = 10;
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.seekPercent()`, `FLVPlayback.totalTime`

FLVPlayback.playheadTime

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.playheadTime

Descripción

Propiedad; número que representa el tiempo o la posición actual (en segundos) de la cabeza lectora, que puede ser un valor fraccionario. Si se establece esta propiedad se activa una búsqueda, con todas las restricciones de una búsqueda.

Cuando cambia el tiempo de la cabeza lectora, lo que ocurre una vez cada 0,25 segundos mientras se reproduce el archivo FLV, el componente distribuye el evento `playheadUpdate`.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se activa hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

En el ejemplo siguiente se capturan instancias del evento `stateChange` que se produce durante la reproducción del archivo FLV y se muestra el tiempo transcurrido de la cabeza lectora en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  my_FLVPlybk.playheadTime);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.playheadUpdate`, `FLVPlayback.playheadUpdateInterval`, `FLVPlayback.seek()`, `FLVPlayback.stateChange`

FLVPlayback.playheadUpdate

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("playheadUpdate", listenerObject);
```

Descripción

Evento; se distribuye mientras se reproduce el archivo FLV con la frecuencia especificada en la propiedad `playheadUpdateInterval`. El valor predeterminado es 0,25 segundos. El componente no distribuye este evento cuando el reproductor de vídeo está en pausa o detenido, a menos que se realice una búsqueda. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`.

Ejemplo

En el ejemplo siguiente se capturan instancias del evento `playheadUpdate` que se produce durante la reproducción del archivo FLV y se muestra el tiempo transcurrido de la cabeza lectora en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdateInterval`

FLVPlayback.playheadUpdateInterval

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.playheadUpdateInterval`

Descripción

Propiedad; número que es la cantidad de tiempo en milisegundos entre cada evento `playheadUpdate`. Si se establece esta propiedad mientras se está reproduciendo el archivo FLV, se reinicia el temporizador. El valor predeterminado es 250.

Dado que los objetos cuepoint de ActionScript se inician cuando se actualiza la cabeza lectora, al disminuir el valor de la propiedad `playheadUpdateInterval`, puede aumentar la precisión de los objetos cuepoint de ActionScript.

Como el intervalo de actualización de la cabeza lectora se establece mediante una llamada a la función global `setInterval()`, la actualización no puede activarse con más frecuencia que la velocidad de fotogramas del archivo SWF, tal y como ocurre con cualquier otro intervalo que se establezca de este modo. Así pues, por ejemplo, para una velocidad de fotogramas predeterminada de 12 fotogramas por segundo, el intervalo efectivo más bajo es aproximadamente 83 milisegundos, o un segundo (1.000 milisegundos) dividido por 12.

Ejemplo

En el ejemplo siguiente se establece el valor de la propiedad `playheadUpdateInterval` en 3000 y se crea un detector que captura instancias del evento `playheadUpdate` a medida que se producen durante la reproducción del archivo FLV. Cuando se produce el evento, el controlador de eventos muestra el tiempo de cabeza lectora transcurrido en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.playheadUpdateInterval = 3000;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  trace("playhead time is: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdate`

FLVPlayback.PLAYING

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.PLAYING
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"playing"`. Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado de reproducción.

Ejemplo

El siguiente ejemplo utiliza la propiedad `FLVPlayback.PLAYING` para comprobar si el estado es `"playing"` cuando se produce un evento `stateChange`. También incluye la constante como parte de un mensaje en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
    if(eventObject.state == FLVPlayback.PLAYING)
      trace(my_FLVPlybk.contentPath + " is now " + FLVPlayback.PLAYING);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.playing

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var :Object = new Object();
listenerObject.playing = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("playing", listenerObject);
```

Descripción

Evento; se distribuye cuando se pasa al estado de reproducción. Esto puede no ocurrir inmediatamente después de que se llame al método `play()` o se haga clic en el control correspondiente; a menudo, se pasa primero al estado de almacenamiento en búfer y después al de reproducción. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

La instancia de FLVPlayback también distribuye el evento `stateChange`.

Ejemplo

En el ejemplo siguiente se muestra el valor de la propiedad `contentPath` en un área de texto cuando se produce el evento `playing`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Arrastre un componente TextArea al escenario, debajo de la instancia de FLVPlayback, y asígnele el nombre de instancia **my_ta**. Añada el siguiente código el fotograma 1 de la línea de tiempo, en el panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playing = function(eventObject:Object):Void {
    my_ta.text = "Now playing: " + my_FLVPlybk.contentPath;
}
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.play(),FLVPlayback.playing, FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.playing

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.playing

Descripción

Propiedad; valor booleano. Es `true` si el archivo FLV está en estado de reproducción. Sólo lectura.

Ejemplo

En el ejemplo siguiente se detectan instancias del evento `stateChange` a medida que se producen durante la reproducción del archivo FLV. Cuando se produce el evento, el ejemplo muestra el valor de la propiedad `playing` en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
trace(my_FLVPlybk.state + ": playing property is " + my_FLVPlybk.playing);
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playing property is " +
  my_FLVPlybk.playing);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.playing, FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.playPauseButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.playPauseButton

Descripción

Propiedad; objeto MovieClip que es PlayPauseButton. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `playPauseButton`, `stopButton`, `backButton` y `forwardButton` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton(**my_plypausebttn**) y StopButton (**my_stopbttn**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV
  PlayPauseButton, StopButton, BackButton y ForwardButton en la Biblioteca
*/
import mx.video.*;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.playButton`, `FLVPlayback.playPauseButton`, `FLVPlayback.paused`, `FLVPlayback.skin`

FLVPlayback.preferredHeight

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.preferredHeight

Descripción

Propiedad; número que especifica la altura del archivo FLV de origen. Esta información no es válida inmediatamente después de llamar a los métodos `play()` o `load()`. Es válida cuando se inicia el evento `ready`. Si el valor de las propiedades `autoSize` o `maintainAspectRatio` es `true`, es preferible leer el valor cuando se inicia el evento `resize`. Sólo lectura.

Ejemplo

El siguiente ejemplo establece el tamaño de la instancia de FLVPlayback cuando se produce el evento `ready`. Cuando se produce el evento `cuePoint`, restablece el tamaño especificado por las propiedades `preferredHeight` y `preferredWidth`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
    trace("width is: " + my_FLVPlybk.width);
    trace("height is: " + my_FLVPlybk.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
listenerObject.ready = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(250, 350);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.addASCuePoint(1.5, "AScp1");
```

Véase también

FLVPlayback.autoSize, FLVPlayback.height, FLVPlayback.maintainAspectRatio, FLVPlayback.preferredWidth, FLVPlayback.ready, FLVPlayback.setSize(), FLVPlayback.setScale(), FLVPlayback.width

FLVPlayback.preferredWidth

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.preferredWidth

Descripción

Propiedad; especifica la anchura del archivo FLV de origen. Esta información no es válida de forma inmediata después de llamar a los métodos `play()` o `load()`, sino cuando se inicia el evento `ready`. Si el valor de las propiedades `autoSize` o `maintainAspectRatio` es `true`, es preferible leer el valor cuando se inicia el evento `resize`. Sólo lectura.

Ejemplo

El siguiente ejemplo establece el tamaño de la instancia de FLVPlayback cuando se produce el evento `ready`. Cuando se produce el evento `cuePoint`, restablece el tamaño especificado por las propiedades `preferredHeight` y `preferredWidth`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
    trace("width is: " + my_FLVPlybk.width);
    trace("height is: " + my_FLVPlybk.height);
};
```

```
my_FLVPlybk.addEventListener("resize", listenerObject);
listenerObject.ready = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(250, 350);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.addASCuePoint(1.5, "AScp1");
```

Véase también

`FLVPlayback.autoSize`, `FLVPlayback.height`, `FLVPlayback.maintainAspectRatio`, `FLVPlayback.preferredHeight`, `FLVPlayback.ready`, `FLVPlayback.setSize()`, `FLVPlayback.setScale()`, `FLVPlayback.width`

FLVPlayback.progress

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("progress", listenerObject);
```

Descripción

Evento; se distribuye con la frecuencia especificada por la propiedad `progressInterval`. Empieza cuando se inicia la carga y finaliza cuando se han cargado todos los bytes o hay un error de red. El valor predeterminado es 0,25 segundos.

Se distribuye sólo para descargas progresivas a través de HTTP. Indica el progreso en la descarga de bytes. El objeto de evento tiene las propiedades `bytesLoaded` y `bytesTotal`, que coinciden con las propiedades de FLVPlayback con los mismos nombres.

El evento tiene además una propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` y `FLVPlayback.visibleVideoPlayerIndex`.

Ejemplo

El siguiente ejemplo establece la propiedad `progressInterval` en 001 milisegundos porque el archivo FLV es pequeño y, a continuación, muestra el número de bytes cargados en cada instancia del evento `progress`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.progressInterval = 001;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    trace(eventObject.bytesLoaded);
}
my_FLVPlybk.addEventListener("progress", listenerObject);
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.addEventListener()`, `FLVPlayback.bytesLoaded`, `FLVPlayback.bytesTotal`, `FLVPlayback.progressInterval`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.progressInterval

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.progressInterval`

Descripción

Propiedad; número que es la cantidad de tiempo en milisegundos entre cada evento `progress`. Si establece el valor de esta propiedad mientras se está reproduciendo el flujo de vídeo, se reinicia el temporizador. El valor predeterminado es 250.

Ejemplo

El siguiente ejemplo establece la propiedad `progressInterval` en 001 milisegundos porque el archivo FLV es pequeño y, a continuación, muestra el número de bytes cargados en cada instancia del evento `progress`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.progressInterval = 001;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    trace(eventObject.bytesLoaded);
}
my_FLVPlybk.addEventListener("progress", listenerObject);
```

Véase también

`FLVPlayback.progress`

FLVPlayback.ready

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("ready", listenerObject);
```

Descripción

Evento; se distribuye cuando el archivo FLV está cargado y preparado para reproducirse. Se inicia la primera vez que se pasa a un estado interactivo después de cargar un nuevo archivo FLV con el método `play()` o `load()`. Sólo se inicia una vez por cada archivo FLV que se cargue.

El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`. La propiedad `vp` es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

Ejemplo

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componente TextArea en el escenario, con el nombre de instancia my_ta
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.autoPlay = false;
my_ta.setSize(260, 30);
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
    my_ta.text = "The FLV is ready. Push Play to start playing";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.addEventListener()`, `FLVPlayback.state`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.removeASCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.removeASCuePoint(CuePoint:Object):Object
my_FLVplybk.removeASCuePoint(time:Number):Object
my_FLVplybk.removeASCuePoint(name:String):Object
```

Parámetros

CuePoint Objeto cuepoint con propiedades `time` y `name` correspondientes al cuepoint que se va a eliminar. El método no comprueba ninguna otra propiedad en el objeto cuepoint entrante. Si el valor de `time` o `name` es `null` o undefined, el método sólo utilizará la propiedad disponible. Si sólo se proporciona el valor de `name`, el método quita el primer cuepoint que tenga este nombre.

time Número que contiene el tiempo del cuepoint que se va a eliminar. El método elimina el primer cuepoint con este tiempo.

name Cadena que contiene el nombre del cuepoint que se va a eliminar. El método elimina el primer cuepoint con este nombre.

Valor devuelto

El objeto cuepoint que se eliminó. Si no hay ningún cuepoint coincidente, el método devuelve `null`.

Descripción

Método; elimina un objeto cuepoint de ActionScript del archivo FLV cargado actualmente. Sólo se usan las propiedades `name` y `time` desde el parámetro *CuePoint* para buscar el cuepoint que se va a eliminar.

Si varios cuepoints de ActionScript coinciden con los criterios de búsqueda, sólo se quitará uno. Para quitarlos todos, llame a esta función repetidamente en un bucle con los mismos parámetros hasta que devuelva null.

La información de cuepoints se elimina cuando se establece la propiedad `contentPath`, por lo que para establecer la información de cuepoints para el siguiente archivo FLV que se va a cargar primero debe establecer la propiedad `contentPath`.

Ejemplo

En el ejemplo siguiente se añade un cuepoint de ActionScript al archivo FLV y después se llama al método `removeASCuePoint()` para eliminarlo. Muestra el nombre del cuepoint eliminado en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
// crear objeto cuepoint
var cuePt:Object = new Object(); // crear objeto cuepoint
var rtn_cuePt:Object = new Object(); // crear objeto para valor devuelto
cuePt.time = 4.444;
cuePt.name = "ripples";
my_FLVPlybk.addASCuePoint(cuePt);  //añadir objeto cuepoint de AS
if ((rtn_cuePt = my_FLVPlybk.removeASCuePoint(cuePt)) != null) {
  trace("Removed cue point: " + rtn_cuePt.name);
}
```

Véase también

`FLVPlayback.addASCuePoint()`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`

FLVPlayback.removeEventListener()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVPlybk.removeEventListener(event:String, listener:Object):Void
my_FLVPlybk.removeEventListener(event:String, listener:Function):Void
```

Parámetros

event Cadena que especifica el nombre del evento para el cual se elimina un detector.

listener Referencia a un objeto detector o a una función que se elimina.

Valor devuelto

Ninguno.

Descripción

Método; elimina un detector de eventos de una instancia de componente.

Ejemplo

El siguiente ejemplo elimina el detector de un evento `cuePoint` cuando se produce el primer objeto cuepoint. Como consecuencia, sólo se detecta el primero de los tres cuepoints.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
Usage 1: listener object
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object(); // crear un objeto detector
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Hit cue point at " + eventObject.info.time);
    my_FLVPlybk.removeEventListener("cuePoint", listenerObject);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

```
Usage 2: listener function
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
function cuePoint(eventObject:Object):Void {
    trace("Hit cue point at " + eventObject.info.time);
    my_FLVPlybk.removeEventListener("cuePoint", cuePoint);
};
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
```

Véase también

`FLVPlayback.addEventListener()`

FLVPlayback.resize

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("resize", listenerObject);
```

Descripción

Evento; se distribuye cuando se cambia el tamaño del vídeo. Esto ocurre cuando se establece la propiedad `visibleVideoPlayerIndex` y se pasa a un reproductor de vídeo con otras dimensiones. El objeto de evento tiene las propiedades `auto`, `x`, `y`, `width`, `height` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica el evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

La propiedad `auto` es `true` cuando el cambio de tamaño es automático porque el valor de la propiedad `autoSize` o `maintainAspectRatio` es `true`. En este caso, es posible que el evento se distribuya para un reproductor de vídeo que no sea el reproductor de vídeo visible. El evento puede distribuirse incuso si las dimensiones no cambian realmente después de intentar cambiar automáticamente el tamaño del componente.

Cuando el valor de la propiedad `auto` es `false`, el evento siempre se aplica al reproductor de vídeo visible. La propiedad `vp` sigue apareciendo, pero siempre será igual a la propiedad `visibleVideoPlayerIndex`.

El componente distribuye el evento (con `auto` establecida en `false`) cuando se establece la propiedad `visibleVideoPlayerIndex` si se cambia el reproductor de vídeo por uno con distintas dimensiones que las del reproductor visible actualmente.

Ejemplo

El siguiente ejemplo reproduce dos archivos FLV. Añade un objeto cuepoint de ActionScript al primer archivo FLV y, cuando se produce el evento `cuePoint`, pasa a un segundo reproductor de vídeo más pequeño para reproducir el segundo archivo FLV. Cuando establece la propiedad `visibleVideoPlayerIndex` para el reproductor de vídeo, activa el evento `resize`, que muestra el tamaño y la ubicación del reproductor de vídeo actual.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// desactivar autoSize y maintainAspectRatio
my_FLVPlybk.autoSize = false;
my_FLVPlybk.maintainAspectRatio = false;
// reproducir este archivo FLV
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
// añadir un objeto cuepoint
my_FLVPlybk.addASCuePoint(3, "switch_here");
var listenerObject:Object = new Object();// crear detector
listenerObject.cuePoint = function(eventObject:Object):Void {
  // añadir un segundo reproductor de vídeo
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  // reproducir este archivo FLV
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // cambiar tamaño de este reproductor de vídeo
  my_FLVPlybk.setSize(240, 180);
  my_FLVPlybk.visibleVideoPlayerIndex = 1; // hacerlo visible
  my_FLVPlybk.play(); // reproducir VLV
};
// añadir detector de evento cuePoint
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
listenerObject.resize = function(eventObject:Object):Void {
  // mostrar ubicación y dimensiones
  trace("Video player is #" + my_FLVPlybk.activeVideoPlayerIndex);
  trace("X coordinate is: " + eventObject.x);
  trace("Y coordinate is: " + eventObject.y);
  trace("Width is: " + eventObject.width);
  trace("Height is: " + eventObject.height);
};
// añadir detector de evento resize
my_FLVPlybk.addEventListener("resize", listenerObject);
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.autoSize, FLVPlayback.height, FLVPlayback.maintainAspectRatio, FLVPlayback.preferredHeight, FLVPlayback.preferredWidth, FLVPlayback.setSize(), FLVPlayback.state, FLVPlayback.width, FLVPlayback.x, FLVPlayback.y

FLVPlayback.rewind

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.rewind = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("rewind", listenerObject);
```

Descripción

Evento; se distribuye cuando se mueve hacia atrás la ubicación de la cabeza lectora mediante una llamada a `seek()` o cuando finaliza la operación de rebobinado automático.

El objeto de evento tiene las propiedades `auto`, `state` y `playheadTime`. Si el evento es el resultado de una búsqueda hacia atrás, el valor de la propiedad `auto` es `false`. Si es el resultado de una operación de rebobinado automático, el valor de la propiedad `auto` es `true`.

La propiedad `playheadTime` indica el tiempo del destino.

El evento `stateChange` se distribuye con un estado `"rewinding"` cuando se produce una operación de rebobinado automático. El evento `stateChange` no se inicia hasta que se ha completado el rebobinado. El evento `seek` se distribuye cuando el rebobinado se produce a través de una búsqueda. La instancia de FLVPlayback también distribuye el evento `playheadUpdate` durante el rebobinado.

El evento `rewind` tiene una propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte las propiedades `FLVPlayback.activeVideoPlayerIndex` y `FLVPlayback.visibleVideoPlayerIndex`.

Ejemplo

El siguiente ejemplo establece la propiedad `autoRewind` en `true` y detecta el evento `rewind`. Cuando se produce el evento, el controlador de eventos muestra los valores de las propiedades `vp`, `state` y `playheadTime` en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoRewind = true;
var listenerObject:Object = new Object();
listenerObject.rewind = function(eventObject:Object) {
  trace("Video player is #" + eventObject.vp);
  trace("State is: " + eventObject.state);
  trace("Playhead time is: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("rewind", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.playheadTime, FLVPlayback.playheadUpdateFLVPlayback.seek(), FLVPlayback.seekPercent(), FLVPlayback.seekSeconds(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.REWINDING

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.REWINDING
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"rewinding"`. Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado de rebobinado.

Ejemplo

El siguiente ejemplo crea un detector del evento `stateChange` y utiliza la propiedad `FLVPlayback.REWINDING` para determinar si el componente está en estado de rebobinado.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.REWINDING)
    trace("The current state is " + FLVPlayback.REWINDING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.scaleX

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.scaleX`

Descripción

Propiedad; número que representa la escala horizontal. La escala estándar es 100.

Ejemplo

En el ejemplo siguiente se establecen las propiedades `scaleX` (horizontal) y `scaleY` (vertical) de la instancia de FLVPlyback al 150%.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.scaleX = 150;
my_FLVPlybk.scaleY = 150;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.setScale()`, `FLVPlayback.scaleY`

FLVPlayback.scaleY

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.scaleY

Descripción

Propiedad; número que representa la escala vertical. La escala estándar es 100.

Ejemplo

En el ejemplo siguiente se establece la escala horizontal (`scaleX`) y la escala vertical (`scaleY`) de la instancia de FLVPlayback al 150%.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.scaleX = 150;
my_FLVPlybk.scaleY = 150;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.scaleX`, `FLVPlayback.setScale()`

FLVPlayback.scrubbing

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.scrubbing`

Descripción

Propiedad; valor booleano. Es `true` si el usuario se desplaza con SeekBar y es `false` en caso contrario. Sólo lectura.

El *desplazamiento* consiste en agarrar el selector de la barra de búsqueda y arrastrarlo en cualquier dirección para buscar una escena determinada del archivo FLV.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `scrubbing` (desplazamiento) cuando se produce un evento `seek`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

NOTA	Debe agarrar el selector de SeekBar, arrastrarlo y soltarlo para producir el evento.

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing)
    trace("User is scrubbing at: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Véase también

FLVPlayback.seek, FLVPlayback.seekBar, FLVPlayback.scrubFinish, FLVPlayback.scrubStart

FLVPlayback.scrubFinish

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.scrubFinish = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("scrubFinish", listenerObject);
```

Descripción

Evento; se distribuye cuando el usuario detiene el desplazamiento en el archivo FLV con SeekBar. El desplazamiento consiste en agarrar el selector de la barra de búsqueda y arrastrarlo en cualquier dirección para buscar una escena determinada del archivo FLV. El desplazamiento se detiene cuando el usuario suelta el selector de SeekBar.

El objeto de evento tiene las propiedades `state` y `playheadTime`. El estado será `"seeking"` hasta después de detener el desplazamiento.

El componente también distribuye el evento `stateChange` con la propiedad state correspondiente al nuevo estado, que debería ser `"playing"`, `"paused"`, `"stopped"` o `"buffering"`.

Ejemplo

El siguiente ejemplo detecta el evento `scrubFinish` y muestra el instante en el que se detiene el desplazamiento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

NOTA	Debe agarrar el selector de SeekBar, arrastrarlo y soltarlo para producir el evento.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.scrubFinish = function(eventObject:Object):Void {
  trace("Scrubbing stopped at " + eventObject.playheadTime);
  trace("Current state is " + eventObject.state);
};
my_FLVPlybk.addEventListener("scrubFinish", listenerObject);
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.seek`, `FLVPlayback.seekBar`, `FLVPlayback.scrubStart`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.scrubStart

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.scrubStart = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("scrubStart", listenerObject);
```

Descripción

Evento; se distribuye cuando el usuario inicia el desplazamiento en el archivo FLV con SeekBar. El desplazamiento consiste en agarrar el selector de SeekBar y arrastrarlo en cualquier dirección para buscar una escena determinada del archivo FLV. El desplazamiento se inicia cuando el usuario hace clic en el selector de SeekBar y finaliza cuando lo suelta.

El objeto de evento tiene las propiedades `state` y `playheadTime`.

El componente también distribuye el evento `stateChange` con la propiedad `state` establecida en "`seeking`". El estado sigue siendo "`seeking`" hasta que el usuario detiene el desplazamiento.

Ejemplo

El siguiente ejemplo detecta el evento `scrubStart` y muestra el instante en el que se inicia el desplazamiento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

NOTA	Debe agarrar el selector de SeekBar y arrastrarlo para producir el evento.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.scrubStart = function(eventObject:Object):Void {
  trace("Scrubbing began at " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("scrubStart", listenerObject);
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.scrubbing`, `FLVPlayback.scrubFinish`, `FLVPlayback.seekBar`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.seek

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("seek", listenerObject);
```

Descripción

Evento; se distribuye cuando se cambia la ubicación de la cabeza lectora al realizar una llamada a `seek()`, establecer la propiedad `playheadTime` o utilizar el control seekBar. La propiedad `playheadTime` indica el tiempo del destino. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica el evento.

La instancia de FLVPlayback distribuye el evento `rewind` cuando la búsqueda es hacia atrás y distribuye el evento `fastForward` cuando la búsqueda es hacia adelante. También distribuye el evento `playheadUpdate`.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El siguiente ejemplo busca el punto en el que hayan transcurrido 2 segundos desde el inicio del archivo FLV cuando se produce el evento `ready`. La función `seek()` activa un evento `seek`, en cuyo punto el detector muestra `playheadTime` y el nombre de la instancia de FLVPlayback.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object) {
        trace("A seek event occurred at " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("seek", listenerObject);
listenerObject.ready = function(eventObject:Object) {
    my_FLVPlybk.seek(2);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.fastForward, FLVPlayback.playheadTime, FLVPlayback.playheadUpdate, FLVPlayback.rewind, FLVPlayback.seek(), FLVPlayback.seekPercent(), FLVPlayback.seekSeconds(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint()

FLVPlayback.seek()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.seek(*time*:Number)

Parámetros

time Número que especifica el tiempo en segundos en que se debe colocar la cabeza lectora.

Valor devuelto

Ninguno.

Descripción

Método; busca un tiempo especificado en el archivo (en segundos, con una precisión de tres decimales).

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El ejemplo siguiente desactiva la reproducción automática del archivo FLV, llama al método `seek()` para mover la cabeza lectora 3 segundos en el vídeo e inicia la reproducción del FLV en ese punto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.seek(3);
my_FLVPlybk.play();
```

Véase también

`FLVPlayback.playheadTime`, `FLVPlayback.seek`, `FLVPlayback.seekPercent()`, `FLVPlayback.seekSeconds()`

FLVPlayback.seekBar

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.seekBar

Descripción

Propiedad; objeto MovieClip que es el control de la barra de búsqueda durante la reproducción. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton`, `stopButton` y `seekBar` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton (**my_plypausbttn**), StopButton (**my_stopbttn**) y SeekBar (**my_seekBar**). Añada las siguientes líneas de código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
  - FLV Custom UI BackButton, ForwardButton, PlayPauseButton, StopButton and
  SeekBar
    components in the Library
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.seekBar = my_seekBar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.scrubbing`, `FLVPlayback.scrubFinish`, `FLVPlayback.scrubStart`, `FLVPlayback.seek`

FLVPlayback.seekBarInterval

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.seekBarInterval

Descripción

Propiedad; número que especifica la frecuencia en milisegundos con que se debe comprobar el selector de la barra de búsqueda durante el desplazamiento. El valor predeterminado es 250.

Dado que este intervalo se establece mediante una llamada a la función global `setInterval()`, la actualización no puede iniciarse con una frecuencia mayor que la velocidad de fotogramas del archivo SWF. Así pues, por ejemplo, para una velocidad de fotogramas predeterminada de 12 fotogramas por segundo, el intervalo efectivo más bajo es aproximadamente 83 milisegundos, o un segundo (1.000 milisegundos) dividido por 12.

Ejemplo

El siguiente ejemplo reduce el valor de `seekBarInterval` a 50 milisegundos y muestra el valor de la propiedad `playheadTime`, si el usuario lleva a cabo un desplazamiento.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
```

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing) {
    my_FLVPlybk.seekBarInterval = 50;
    trace("User is scrubbing at: " + eventObject.playheadTime);
  }
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Véase también

`FLVPlayback.seekBar`, `FLVPlayback.seekBarScrubTolerance`

FLVPlayback.seekBarScrubTolerance

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.seekBarScrubTolerance`

Descripción

Propiedad; número que especifica hasta dónde puede mover un usuario el selector de SeekBar antes de que se produzca una actualización. El valor se especifica como un porcentaje (entre 1 y 100). El valor predeterminado es 5.

Ejemplo

El siguiente ejemplo comprueba si el usuario lleva a cabo un desplazamiento cuando se produce un evento `seek` y, de ser así, reduce el valor de la propiedad `seekBarScrubTolerance` a 0 para aumentar la actualización de la ubicación de SeekBar y la frecuencia del evento `seek`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

NOTA	Debe agarrar el selector de SeekBar, arrastrarlo y soltarlo para producir el evento.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing) {
    my_FLVPlybk.seekBarScrubTolerance = 0;
    trace("User is scrubbing at: " + eventObject.playheadTime);
  }
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Véase también

`FLVPlayback.scrubbing`, `FLVPlayback.scrubFinish`, `FLVPlayback.scrubStart`, `FLVPlayback.seekBar`, `FLVPlayback.seekBarInterval`

FLVPlayback.SEEKING

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.SEEKING
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena "`seeking`". Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado de búsqueda.

Ejemplo

El siguiente ejemplo utiliza la propiedad `FLVPlayback.SEEKING` para comprobar si el estado es `"seeking"` cuando se produce un evento `stateChange`. Si es así, muestra un mensaje donde se indica.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.SEEKING)
    trace("The current state is " + FLVPlayback.SEEKING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.seekPercent()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`my_FLVplybk``.seekPercent(`*`percent`*`:Number)`

Parámetros

`percent` Número que especifica un porcentaje de la duración del archivo FLV en el que se colocará la cabeza lectora.

Valor devuelto

Ninguno.

Descripción

Método; busca un punto correspondiente a un porcentaje del archivo y coloca allí la cabeza lectora. El porcentaje es un número entre 0 y 100.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El siguiente ejemplo desactiva la reproducción automática del archivo FLV. Cuando el archivo FLV está listo, establece la cabeza lectora en un punto correspondiente al 30% del tiempo de reproducción e inicia la reproducción en ese punto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekPercent(30);
  my_FLVPlybk.play();
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`FLVPlayback.seek`, `FLVPlayback.seek()`, `FLVPlayback.seekSeconds()`

FLVPlayback.seekSeconds()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.seekSeconds(*time*:Number)

Parámetros

time Número que especifica el tiempo en segundos con respecto al tiempo total de reproducción en que se debe colocar la cabeza lectora.

Valor devuelto

Ninguno.

Descripción

Método; busca un tiempo especificado en el archivo en segundos, con una precisión de hasta tres decimales (milisegundos). Este método realiza la misma operación que el método `seek()`; se proporciona por simetría con el método `seekPercent()`.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El ejemplo siguiente desactiva la reproducción automática del archivo FLV, llama al método `seekSeconds()` para mover la cabeza lectora 5 segundos en el vídeo e inicia la reproducción del FLV en ese punto.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekSeconds(4);
  my_FLVPlybk.play();
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Véase también

`FLVPlayback.seek`, `FLVPlayback.seek()`, `FLVPlayback.seekPercent()`

FLVPlayback.seekToNavCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.seekToNavCuePoint(time:Number):Void
my_FLVplybk.seekToNavCuePoint(name:String):Void
my_FLVplybk.seekToNavCuePoint(cuePoint:Object):Void
```

Parámetros

time Número que indica el tiempo del cuepoint de navegación que se va a buscar. El método sólo utiliza los primeros tres decimales y redondea los decimales adicionales.

name Cadena que contiene el nombre del cuepoint que se va a buscar.

cuePoint Objeto cuepoint en el que se establecen las propiedades `time` y `name` para especificar el cuepoint que se va a buscar.

Valor devuelto

Ninguno.

Descripción

Método; busca un cuepoint de navegación que coincida con el tiempo especificado o uno posterior. Si el valor de time es undefined, `null` o menor que 0, el método inicia la búsqueda en el tiempo equivalente a 0.

Si sólo se especifica un valor de tiempo, el método busca un cuepoint que coincida con ese tiempo o uno posterior.

Si se especifica un nombre, el método busca el primer cuepoint activado que coincida con dicho nombre. Para más información sobre cómo activar y desactivar cuepoints, consulte "FLVPlayback.setFLVCuePointEnabled()" en la página 690.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El siguiente ejemplo busca el cuepoint denominado point2 cuando se produce el evento `ready`. El controlador de eventos `cuePoint` muestra los valores de `name`, `time` y `type` para cada cuepoint que tenga lugar.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekToNavCuePoint("point2");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point time is: " + eventObject.info.time);
    trace("Cue point type is: " + eventObject.info.type);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.cuePoint`, `FLVPlayback.seek`, `FLVPlayback.seek()`, `FLVPlayback.seekToNextNavCuePoint()`

FLVPlayback.seekToNextNavCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.seekToNextNavCuePoint([*time*:Number])

Parámetros

time Número que es el tiempo de inicio (en segundos) desde el que se buscará el siguiente cuepoint de navegación. El valor predeterminado es la propiedad `playheadTime` actual. Opcional.

Valor devuelto

Ninguno.

Descripción

Método; busca el siguiente cuepoint de navegación, en función del valor actual de la propiedad `playheadTime`. El método salta los cuepoints de navegación que se hayan desactivado y pasa al final del archivo FLV si no hay ningún otro cuepoint.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El siguiente ejemplo busca el siguiente cuepoint de navegación cuando se produce el cuepoint denominado `point2`. Como consecuencia, se salta esa parte del archivo FLV entre los cuepoints point2 y point3.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    if(eventObject.info.name == "point2")
      my_FLVPlybk.seekToNextNavCuePoint(eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

FLVPlayback.cuePoint, FLVPlayback.findCuePoint(), FLVPlayback.playheadTime, FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.seekToPrevNavCuePoint()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.seekToPrevCueNavPoint([time:Number])
```

Parámetros

time Número que es el tiempo de inicio (en segundos) desde el que se buscará el anterior cuepoint de navegación. El valor predeterminado es el valor actual de la propiedad `playheadTime`. Opcional.

Valor devuelto

Ninguno.

Descripción

Método; busca el anterior cuepoint de navegación, en función del valor actual de la propiedad `playheadTime`. Va al principio si no hay un cuepoint anterior. Este método salta los cuepoints de navegación que hayan sido desactivados.

Por varias razones, es posible que la propiedad `playheadTime` no tenga el valor esperado inmediatamente después de llamar a uno de los métodos de búsqueda o de establecer `playheadTime` para provocar la búsqueda. En primer lugar, en una descarga progresiva, sólo se puede buscar en un fotograma clave, de forma que la búsqueda devuelve como resultado el tiempo del primer fotograma clave después del tiempo especificado. (En la transmisión de flujo, una búsqueda siempre devuelve el tiempo exacto especificado aunque el archivo FLV de origen no tenga ningún fotograma clave en tal punto.) En segundo lugar, la búsqueda es asíncrona, de forma que si se llama a un método de búsqueda o se establece la propiedad `playheadTime`, el valor de `playheadTime` no se actualiza inmediatamente. Para obtener el tiempo después de completar la búsqueda, detecte el evento `seek`, que no se inicia hasta que se actualiza la propiedad `playheadTime`.

Ejemplo

El siguiente ejemplo busca el cuepoint de navegación anterior cuando se produce el cuepoint point2 y crea un bucle para reproducir el archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    if(eventObject.info.name == "point2")
      my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Véase también

FLVPlayback.cuePoint, FLVPlayback.findCuePoint(), FLVPlayback.playheadTime, FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.seekToPrevOffset

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.seekToPrevOffset

Descripción

Propiedad; número (de segundos) que utiliza el método `seekToPrevNavCuePoint()` cuando compara su tiempo con el del cuepoint anterior. El método utiliza esta valor para garantizar que, si está justo a continuación de un cuepoint, puede saltar al anterior sin tener que ir al mismo cuepoint que acaba de producirse. El valor predeterminado es un segundo.

Ejemplo

El siguiente ejemplo establece inicialmente la propiedad `seekToPrevOffset` en 10, de modo que la primera llamada al método `seekToPrevNavCuePoint()` se dirige al cuepoint denominado point1. Sin embargo, cuando se produce el evento `cuePoint` inicial de point3, el ejemplo reduce el valor de la propiedad `seekToPrevOffset` a 1 segundo, de modo que las llamadas posteriores al método `seekToPrevNavCuePoint()` se dirigen al cuepoint denominado point2.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekToPrevOffset = 10;
  my_FLVPlybk.seekToNavCuePoint("point3");
}
my_FLVPlybk.addEventListener("ready", listenerObject)
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    trace("hit cue point at " + eventObject.info.time);
    if(eventObject.info.name == "point3"){
      my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time);
      my_FLVPlybk.seekToPrevOffset = 1;
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject)
```

Véase también

`FLVPlayback.seekToPrevNavCuePoint()`

FLVPlayback.setFLVCuePointEnabled()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, time:Number)
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, name:String)
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, cuePoint:Object)
```

Parámetros

enabled Valor booleano que especifica si se debe activar (`true`) o desactivar (`false`) un cuepoint del archivo FLV.

time Número que representa el tiempo (en segundos) del cuepoint que se va a establecer.

name Nombre del cuepoint que se va a establecer.

cuePoint Objeto cuepoint con propiedades `name` y `time` correspondientes al cuepoint que se va a establecer. El método no comprueba ninguna otra propiedad en el objeto cuepoint entrante. Si el valor de `time` o `name` no está definido, el método intenta buscar un cuepoint utilizando únicamente el valor disponible.

Valor devuelto

Un número. Si el valor de `metadataLoaded` es `true`, el método devuelve el número de cuepoints cuyo estado de activación ha cambiado. Si `metadataLoaded` es `false`, el método devuelve –1 porque el componente no puede determinar todavía los cuepoints que se van a establecer, si es que hay alguno. Sin embargo, cuando se reciben los metadatos, el componente establece los cuepoints especificados de la forma correspondiente.

Descripción

Método; activa o desactiva uno o más cuepoints de archivos FLV. Los cuepoints desactivados se desactivan para distribuirlos como eventos y para navegar hasta ellos con los métodos `seekToPrevNavCuePoint()`, `seekToNextNavCuePoint()` y `seekToNavCuePoint()`.

La información de cuepoint se elimina cuando se establece la propiedad `contentPath` en un archivo FLV distinto. Así pues, establezca la propiedad `contentPath` antes de establecer la información de cuepoint para el siguiente archivo FLV que se va a cargar.

Los cambios provocados por esta función no se reflejan mediante llamadas al método `isFLVCuePointEnabled()` hasta que se cargan los metadatos.

Ejemplo

El siguiente ejemplo desactiva los cuepoints point2 y point3 cuando se produce el evento `ready`. El controlador de eventos `cuePoint` muestra en el panel Salida el nombre y el tiempo de cada cuepoint que se produce. El archivo FLV contiene los siguientes cuepoints incorporados: point1 en 00:00:00:418; point2 en 00:00:07.748; point3 en 00:00:16:020.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
function ready(eventObject:Object) {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
  my_FLVPlybk.setFLVCuePointEnabled(false, 16.02);
}
my_FLVPlybk.addEventListener("ready", ready);
function cuePoint(eventObject:Object) {
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point time is: " + eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.cuePoint`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`

FLVPlayback.setScale()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
my_FLVplybk.setScale(xs:Number, ys:Number)
```

Parámetros

xs Número que representa la escala horizontal.

ys Número que representa la escala vertical.

Valor devuelto

Ninguno.

Descripción

Método; establece simultáneamente los valores de las propiedades `scaleX` y `scaleY`. Si se establecen por separado los valores de las propiedades `scaleX` y `scaleY` podría producirse un cambio automático de tamaño, por lo que resulta más eficaz establecer los valores simultáneamente.

Si el valor de `autoSize` es `true`, este método no produce ningún efecto, ya que el reproductor establece sus propias dimensiones. Si el valor de la propiedad `maintainAspectRatio` es `true` y el valor de `autoSize` es `false`, al cambiar los valores de `scaleX` o `scaleY` se provoca un cambio de tamaño automático.

Ejemplo

En el ejemplo siguiente se llama al método `setScale()` para ajustar la escala de las dimensiones horizontal (*x*) y vertical (*y*) de la instancia de FLVPlayback. El ejemplo establece el valor de la propiedad `maintainAspectRatio` en `false` para evitar el cambio automático de tamaño y permitir utilizar la escala especificada.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;

my_FLVPlybk.maintainAspectRatio = false;
my_FLVPlybk.setScale(200, 175);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.scaleX`, `FLVPlayback.scaleY`, `FLVPlayback.setSize()`

FLVPlayback.setSize()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.setSize(*w*:Number, *h*:Number)

Parámetros

w Número que especifica la anchura del reproductor de vídeo.

h Número que especifica la altura del reproductor de vídeo.

Valor devuelto

Ninguno.

Descripción

Método; establece simultáneamente la altura y la anchura. Si se establecen por separado los valores de las propiedades `width` y `height` podría producirse un cambio automático de tamaño, por lo que resulta más eficaz establecer los valores simultáneamente.

Si el valor de `autoSize` es `true`, este método no produce ningún efecto, ya que el reproductor establece sus propias dimensiones. Si el valor de la propiedad `maintainAspectRatio` es `true` y el valor de `autoSize` es `false`, al cambiar los valores de anchura o altura se provoca un cambio de tamaño automático.

Ejemplo

El ejemplo siguiente llama al método `setSize()` para establecer el tamaño de la instancia de FLVPlayback a una anchura de 150 píxeles y una altura de 150 píxeles. El controlador de eventos `resize` muestra la anchura y altura reales porque el valor predeterminado de la propiedad `maintainAspectRatio` es `true`, de modo que se mantiene la proporción al cambiar automáticamente el tamaño.

Arrastre el componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
```

```
// el valor predeterminado de maintainAspectRatio es true; las dimensiones
  lo reflejarán
my_FLVPlybk.setSize(150, 150);
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
  trace("Player's width is: " + my_FLVPlybk.width)
  trace("Player's height is: " + my_FLVPlybk.height)
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.height, FLVPlayback.width, FLVPlayback.setScale()

FLVPlayback.skin

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.skin

Descripción

Propiedad; cadena que especifica la URL de un archivo SWF de aspecto. Esta cadena puede contener un nombre de archivo, una ruta relativa, como Skins/my_Skin.swf, o una URL absoluta, como http://www.myskins.org/MySkin.swf.

Ejemplo

En el ejemplo siguiente se aplica el aspecto ArcticExternal.swf a una instancia del componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Copie el archivo ArcticExternalAll.swf de la carpeta Configuration/Skins de Flash a su carpeta de trabajo. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
my_FLVPlybk.skin = "ArcticExternalAll.swf";
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.bufferingBarHidesAndDisablesOthers`, `FLVPlayback.skinAutoHide`, `FLVPlayback.skinError`, `FLVPlayback.skinLoaded`

FLVPlayback.skinAutoHide

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

my_FLVplybk.skinAutoHide

Descripción

Propiedad; valor booleano. Si es `true`, oculta el aspecto del componente cuando el ratón no está sobre el vídeo. Esta propiedad sólo afecta a los aspectos que se cargan al establecer la propiedad `skin` y no a un aspecto que cree desde los componentes de interfaz de usuario personalizados de reproducción de FLV. El valor predeterminado es `false`.

Ejemplo

El siguiente ejemplo establece la propiedad `skinAutoHide` en `true`, de modo que el aspecto del componente, que incluye los controles de reproducción, no aparece a menos que el puntero del ratón se desplace sobre el vídeo.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Seleccione un aspecto en el inspector de componentes. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
my_FLVPlybk.skinAutoHide = true;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.bufferingBarHidesAndDisablesOthers`, `FLVPlayback.skin`

FLVPlayback.skinError

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.skinError = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("skinError", listenerObject);
```

Descripción

Evento; se distribuye cuando se produce un error al cargar un archivo SWF de aspecto. El evento tiene una propiedad `message` que contiene el mensaje de error.

Ejemplo

En el siguiente ejemplo, se intenta cargar la propiedad `skin` con el nombre de un archivo de aspecto ficticio y se muestra el contenido de la propiedad `message` del evento cuando se produce el evento `skinError`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. A continuación, añada el siguiente código al fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.skinError = function(eventObject:Object):Void {
  trace(eventObject.message);
}
my_FLVPlybk.addEventListener("skinError", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
my_FLVPlybk.skin = "NoSuchSkin.swf";
```

Véase también

`FLVPlayback.skin`, `FLVPlayback.skinLoaded`

FLVPlayback.skinLoaded

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.skinLoaded = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("skinLoaded", listenerObject);
```

Descripción

Evento; se distribuye cuando se carga un archivo SWF de aspecto. El componente no inicia la reproducción de un archivo FLV hasta que se han iniciado los eventos `ready` y `skinLoaded` (o `skinError`).

Ejemplo

El siguiente ejemplo muestra el nombre del aspecto del componente cuando se inicia el evento `skinLoaded`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. A continuación, añada el siguiente código al fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.skinLoaded = function(eventObject:Object):Void {
  trace("Skin: " + eventObject.target.skin + " has loaded");
};
my_FLVPlybk.addEventListener("skinLoaded", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.skin`, `FLVPlayback.skinError`

FLVPlayback.state

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.state`

Descripción

Propiedad; cadena que especifica el estado del componente. Esta propiedad se establece con los métodos `load()`, `play()`, `stop()`, `pause()` y `seek()`. Sólo lectura.

Los valores posibles de la propiedad `state` son: `"buffering"`, `"connectionError"`, `"disconnected"`, `"loading"`, `"paused"`, `"playing"`, `"rewinding"`, `"seeking"` y `"stopped"`. Puede utilizar las propiedades de la clase FLVPlayback para probar estos estados. Para más información, consulte “Propiedades de la clase FLVPlayback” en la página 559.

Ejemplo

El siguiente ejemplo muestra la propiedad `state` en el panel Salida cada vez que se produce el evento `stateChange` durante la reproducción del archivo.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.buffering`, `FLVPlayback.paused`, `FLVPlayback.playing`, `FLVPlayback.stateChange`, `FLVPlayback.stopped`

FLVPlayback.stateChange

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("stateChange", listenerObject);
```

Descripción

Evento; se distribuye cuando cambia el estado de reproducción. El objeto de evento tiene las propiedades `state` y `playheadTime`.

Se puede utilizar este evento para hacer un seguimiento cuando la reproducción pasa a (o deja de estar en) un estado no interactivo (por ejemplo, durante una conexión, al cambiar de tamaño o al rebobinar) en los que los métodos `play()`, `pause()`, `stop()` y `seek()` colocarán en cola las peticiones que se deben ejecutar cuando el reproductor pase a un estado interactivo.

El evento tiene una propiedad `vp`, que es el número de índice del reproductor de vídeo al que se aplica este evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

Ejemplo

El siguiente ejemplo muestra la propiedad `state` en el panel Salida cada vez que se produce el evento `stateChange` durante la reproducción del archivo.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.state`

FLVPlayback.stateResponsive

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.stateResponsive`

Descripción

Propiedad; valor booleano. Es `true` si el estado es interactivo. Si el estado no es interactivo, las llamadas a los métodos `play()`, `load()`, `stop()`, `pause()` y `seek()` se colocarán en cola y se ejecutarán más tarde, cuando se pase a un estado interactivo. Dado que las llamadas se colocan en cola y se ejecutan más tarde, normalmente no es necesario realizar un seguimiento del valor de la propiedad `stateResponsive`. Los estados interactivos son: `disconnected`, `stopped`, `playing`, `paused` y `buffering`. Sólo lectura.

Ejemplo

El siguiente ejemplo muestra los valores de las propiedades `state` y `stateResponsive` cuando el estado cambia durante la reproducción del archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + "; responsive: " +
  my_FLVPlybk.stateResponsive);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.stop()

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`my_FLVPlybk``.stop()`

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; detiene la reproducción del vídeo. Si el valor de la propiedad `autoRewind` es `true`, el archivo FLV se rebobina hasta el principio.

Ejemplo

En el ejemplo siguiente se detecta el evento `playheadUpdate` y cuando el valor de `playheadTime` transcurrido es igual o mayor que 5 segundos, el detector llama al método `stop()` para detener la reproducción del archivo FLV. Un segundo detector detecta el evento `stopped` y muestra los valores de las propiedades `playheadTime` y `state`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoRewind = false;
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  trace("playhead time is: " + eventObject.playheadTime);
  trace("The video player state is: " + eventObject.state);
};
my_FLVPlybk.addEventListener("stopped", listenerObject);
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if (eventObject.playheadTime >= 5) {
    my_FLVPlybk.stop();
  }
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath ="http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.pause()`, `FLVPlayback.play()`

FLVPlayback.stopButton

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.stopButton

Descripción

Propiedad; objeto MovieClip que es el control del botón de parada. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton` y `stopButton` para asociar controles individuales de interfaz de usuario personalizados de reproducción FLV a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de reproducción FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) y StopButton (**my_stopbttn**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
  - FLV Custom UI BackButton, ForwardButton, PlayButton, PauseButton, and
    StopButton components in the Library
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.pauseButton, FLVPlayback.playButton, FLVPlayback.playPauseButton, FLVPlayback.skin, FLVPlayback.stopped

FLVPlayback.STOPPED

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.STOPPED
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene la constante de cadena `"stopped"`. Puede comparar esta propiedad con la propiedad `state` para determinar si el componente está en estado detenido.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `FLVPlayback.STOPPED` cuando el componente pasa al estado detenido.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.STOPPED)
    trace("State is " + FLVPlayback.STOPPED);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.stopped

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("stopped", listenerObject);
```

Descripción

Evento; se distribuye cuando se pasa al estado detenido. Esto ocurre al llamar al método `stop()` o al hacer clic en el control `stopButton`. También sucede en algunos casos si la propiedad `autoPlay` es `false` (se podría pasar al estado `paused`) cuando se carga el archivo FLV. La instancia de FLVPlayback también distribuye este evento cuando la cabeza lectora se detiene al final del archivo FLV. El objeto de evento tiene las propiedades `state`, `playheadTime` y `vp`, que es el número de índice del reproductor de vídeo al que se aplica el evento. Para más información sobre la propiedad `vp`, consulte `FLVPlayback.activeVideoPlayerIndex` en la página 567 y `FLVPlayback.visibleVideoPlayerIndex` en la página 711.

La instancia de FLVPlayback también distribuye el evento `stateChange`.

Ejemplo

En el ejemplo siguiente se detectan instancias del evento `stopped` a medida que se producen durante la reproducción del archivo FLV. Cuando se produce el evento, el ejemplo muestra el tiempo de cabeza lectora transcurrido en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("stopped", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.playheadTime`, `FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stop()`

FLVPlayback.stopped

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.stopped`

Descripción

Propiedad; valor booleano. Es `true` si el estado de la instancia de FLVPlayback es `stopped`. Sólo lectura.

Ejemplo

En el ejemplo siguiente se detectan instancias del evento `stateChange` a medida que se producen durante la reproducción del archivo FLV. Cuando se produce el evento, el ejemplo muestra el valor de la propiedad `stopped` en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": stopped property is: " +
  my_FLVPlybk.stopped);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stop()`, `FLVPlayback.stopped`

FLVPlayback.totalTime

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.totalTime`

Descripción

Propiedad; número que representa el tiempo total de reproducción del vídeo en segundos. Al transmitir desde un servidor FCS utilizando el objeto NCManager predeterminado, este valor se determinará automáticamente mediante las API del servidor y sustituirá los valores establecidos mediante esta propiedad o recopilados de los metadatos. También es true si se establece este valor en un archivo SMIL. Se podrá leer el valor de la propiedad cuando se pase al estado detenido o de reproducción tras establecer el valor de la propiedad `contentPath`. Esta propiedad no se utiliza para flujos dinámicos transmitidos desde un servidor FCS.

En la descarga HTTP, el valor se determina automáticamente si el archivo FLV tiene metadatos incorporados; de lo contrario, se establece explícitamente o es 0. Si se establece explícitamente, se omitirá el valor de los metadatos del flujo.

Cuando se establece el valor de esta propiedad, dicho valor se aplicará al siguiente archivo FLV que se cargue estableciendo `contentPath`. No afecta a los archivos FLV que ya se hayan cargado. Además, esta propiedad no devuelve el nuevo valor pasado hasta que se carga un archivo FLV.

Si no se establece esta propiedad (de forma explícita o automáticamente), la reproducción seguirá funcionando pero podrían producirse problemas con los controles de búsqueda.

Ejemplo

En el ejemplo siguiente se muestra el tiempo total (en segundos) del archivo FLV cuando se produce el evento `ready`, una vez completada la carga.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  trace("Total play time for this video is: " + my_FLVPlybk.totalTime);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.contentPath`, `FLVPlayback.playheadTime`, `FLVPlayback.playing`, `FLVPlayback.stopped`

FLVPlayback.transform

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.transform

Descripción

Propiedad; objeto que ofrece acceso directo a los métodos `Sound.setTransform()` y `Sound.getTransform()` para proporcionar control del sonido. Debe establecer esta propiedad en un objeto para inicializarlo y para que los cambios surtan efecto. Al leer el valor de la propiedad obtiene una copia de la configuración actual, que puede modificar. El valor predeterminado es `undefined`.

Ejemplo

El siguiente ejemplo establece la propiedad `transform` para reproducir el sonido del archivo FLV únicamente en el altavoz izquierdo.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
/* Reproducir todo el audio sólo en el altavoz izquierdo */
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object) {
  if (eventObject.target.state == "loading") {  // si se carga
    myTransform = new Object();
    myTransform.ll = 100;
    myTransform.lr = 100;
    myTransform.rr = 0;
    myTransform.rl = 0;
    my_FLVPlybk.transform = myTransform;
  }
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.volume`

FLVPlayback.version

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

```
mx.video.FLVPlayback.version
```

Descripción

Propiedad de sólo lectura de la clase FLVPlayback que contiene el número de versión del componente.

Ejemplo

El siguiente ejemplo muestra el número de versión del componente en el panel Salida. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.video.*;
trace(FLVPlayback.version);
```

FLVPlayback.visible

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk`.visible`

Descripción

Propiedad; valor booleano. Si es `true`, hace que el componente FLVPlayback sea visible. Si es `false`, hace que el componente sea invisible. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se establece el valor de la propiedad `visible` en `false` para hacer que la instancia de FLVPlayback sea invisible cuando haya acabado la reproducción del archivo FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  my_FLVPlybk.visible = false;
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.closeVideoPlayer()`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.visibleVideoPlayerIndex

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.visibleVideoPlayerIndex`

Descripción

Propiedad; número que puede utilizar para administrar varios flujos de archivos FLV. Establece la instancia del reproductor de vídeo que puede verse, oírse y controlarse mediante el aspecto o los controles de reproducción, mientras los demás reproductores de vídeo quedan ocultos y silenciados. El valor predeterminado es 0. No convierte al reproductor de vídeo en el destino de la mayoría de las API; utilice la propiedad `activeVideoPlayerIndex` en su lugar.

Los métodos y las propiedades que controlan las dimensiones interactúan con esta propiedad. Los métodos y las propiedades que establecen las dimensiones del reproductor de vídeo (`setScale()`, `setSize()`, `width`, `height`, `scaleX`, `scaleY`) pueden utilizarse en todos los reproductores de vídeo. Sin embargo, podrían tener dimensiones diferentes, dependiendo de si el valor de `autoSize` o `maintaintAspectRatio` está establecido en esos reproductores de vídeo. Al leer las dimensiones a través de las propiedades `width`, `height`, `scaleX` y `scaleY` obtendrá las dimensiones del reproductor de vídeo visible únicamente. Los otros reproductores de vídeo pueden tener (o no) las mismas dimensiones.

Para obtener las dimensiones de diversos reproductores de vídeo cuando no están visibles hay que detectar el evento `resize` y almacenar el valor del tamaño.

Esta propiedad no tiene ninguna implicación para la visibilidad de todo el componente, sino sólo para el reproductor de vídeo que está visible cuando el componente está visible. Para establecer la visibilidad de todo el componente, utilice la propiedad `visible`.

Ejemplo

El siguiente ejemplo crea dos reproductores de vídeo para reproducir dos archivos FLV de forma consecutiva en una sola instancia de FLVPlayback. Establece la propiedad `visibleVideoPlayerIndex` para hacer visibles los reproductores de vídeo y los archivos FLV.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// especificar nombre y ubicación del archivo FLV correspondiente al
  reproductor predeterminado
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv"
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // añadir un segundo reproductor de vídeo y especificar el nombre y la
  ubicación de su archivo FLV
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // restablecer el reproductor de vídeo predeterminado, que reproduce su
  archivo FLV automáticamente
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
```

```
listenerObject.complete = function(eventObject:Object):Void {
  // si complete hace referencia al segundo archivo FLV, convertir al
  predeterminado en activo y visible
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
  } else { // convertir al segundo reproductor en activo y visible y
  reproducir archivo FLV
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// añadir detector de evento complete
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Véase también

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.setScale(), FLVPlayback.setSize(), FLVPlayback.height, FLVPlayback.width, FLVPlayback.scaleX, FLVPlayback.scaleY, FLVPlayback.visible

FLVPlayback.volume

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.volume

Descripción

Propiedad; número del intervalo 0 a 100 que indica el nivel del control de volumen. El valor predeterminado es 100.

Ejemplo

En el ejemplo siguiente se establece el volumen inicial en un valor relativamente bajo: 10.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// Puede cambiar este valor de 0 a 100
my_FLVPlybk.volume = 10;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.transform`, `FLVPlayback.volumeBar`, `FLVPlayback.volumeBarInterval`, `FLVPlayback.volumeUpdate`

FLVPlayback.volumeBar

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.volumeBarInterval`

Descripción

Propiedad; objeto MovieClip que es el control de la barra de volumen. Para más información sobre el uso de componentes de interfaz de usuario personalizados de reproducción de FLV para controlar la reproducción, consulte "Aplicación de aspectos a componentes individuales de interfaz de usuario personalizados de reproducción FLV" en la página 543.

Ejemplo

En el ejemplo siguiente se utilizan las propiedades `backButton`, `forwardButton`, `playButton`, `pauseButton`, `stopButton` y `volumeBar` para asociar controles personalizados de interfaz de usuario de FLV individuales a un componente FLVPlayback.

Arrastre un componente FLVPlayback al escenario, asígnele el nombre de instancia **my_FLVPlybk** y establezca el parámetro `skin` en None en el inspector de componentes. A continuación, añada los siguientes componentes de interfaz de usuario personalizados de FLV individuales y asígneles los nombres de instancia mostrados entre paréntesis: BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**), StopButton (**my_stopbttn**) y VolumeBar (**my_vBar**). Después añada las siguientes líneas de código al panel Acciones:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
  - Componentes personalizados de interfaz de usuario de FLV BackButton,
  ForwardButton, PlayButton, PauseButton, StopButton y VolumeBar en la
  Biblioteca
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.volumeBar = my_vBar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.volume`, `FLVPlayback.volumeBarInterval`, `FLVPlayback.volumeBarScrubTolerance`, `FLVPlayback.volumeUpdate`

FLVPlayback.volumeBarInterval

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.volumeBarInterval

Descripción

Propiedad; número que especifica la frecuencia en milisegundos con que se debe comprobar la ubicación del selector de la barra de volumen durante el desplazamiento. El valor predeterminado es 250.

Ejemplo

El siguiente ejemplo establece la propiedad `volumeBarInterval` en 1 segundo (1000 milisegundos) y crea un evento `volumeUpdate` que muestra el tiempo de la cabeza lectora y el volumen mientras el usuario arrastra el selector en la barra de volumen. El evento `volumeUpdate` se produce aproximadamente a intervalos de 1 segundo como consecuencia del valor establecido en `volumeBarInterval`.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.volumeBarInterval = 1000;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object) {
    trace("Playhead time is: " + my_FLVPlybk.playheadTime);
    trace("Volume is: " + my_FLVPlybk.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.volume`, `FLVPlayback.volumeBar`, `FLVPlayback.volumeBarScrubTolerance`, `FLVPlayback.volumeUpdate`

FLVPlayback.volumeBarScrubTolerance

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.volumeBarScrubTolerance

Descripción

Propiedad; número que especifica hasta dónde puede mover un usuario el selector de la barra de volumen antes de que se produzca una actualización. El valor se expresa como un porcentaje. El valor predeterminado es 5.

Ejemplo

El siguiente ejemplo establece la propiedad `volumeBarScrubTolerance` en `20` y crea un evento `volumeUpdate` que muestra el volumen establecido mientras el usuario arrastra el selector en la barra de volumen.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.volumeBarScrubTolerance = 20;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object) {
    trace("Playhead time is: " + my_FLVPlybk.playheadTime);
    trace("Volume is: " + my_FLVPlybk.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.volumeBar`, `FLVPlayback.volumeBarInterval`

FLVPlayback.volumeUpdate

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object):Void {
  // insertar aquí código de gestión de eventos
};
my_FLVplybk.addEventListener("volumeUpdate", listenerObject);
```

Descripción

Evento; se distribuye cuando el volumen cambia porque el usuario mueve el selector del control volumeBar o porque se establece la propiedad `volume` en ActionScript. El objeto de evento tiene una propiedad `volume`.

Ejemplo

El siguiente ejemplo muestra el valor de la propiedad `volume` en el panel Salida con los ajustes que el usuario hace al volumen.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object):Void {
  trace("Volume setting is: " + eventObject.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Véase también

`FLVPlayback.addEventListener()`, `FLVPlayback.volume`, `FLVPlayback.volumeBar`

FLVPlayback.width

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.width

Descripción

Propiedad; número que especifica la anchura de la instancia de FLVPlayback en el escenario. Esta propiedad sólo afecta a la anchura de la instancia de FLVPlayback y no incluye la anchura de un archivo SWF de aspecto que pueda cargarse. Utilice la propiedad `width` de FLVPlayback y no la propiedad `MovieClip._width` porque la propiedad `_width` podría devolver un valor distinto si se carga un archivo SWF de aspecto.

Ejemplo

En el ejemplo siguiente se establecen los valores de las propiedades `width` y `height`, lo que activa un evento `resize`, ya que el valor predeterminado de la propiedad `maintainAspectRatio` es `true`. Cuando se produce el evento, el controlador de eventos muestra la anchura y la altura de la instancia de FLVPlayback a la que se ha cambiado el tamaño en el panel Salida.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
// maintainAspectRatio (true de forma predeterminada) afecta a las
  dimensiones reales
my_FLVPlybk.width = 400;
my_FLVPlybk.height = 350;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  trace("Width is: " + eventObject.target.width + " Height is: " +
  eventObject.target.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Véase también

`FLVPlayback.height`, `FLVPlayback.setSize()`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`

FLVPlayback.x

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

`my_FLVPlybk``.x`

Descripción

Propiedad; número que especifica la ubicación de la coordenada horizontal del reproductor de vídeo. Esta propiedad sólo afecta a la ubicación horizontal de la instancia de FLVPlayback y no incluye la ubicación de un archivo SWF de aspecto que pueda modificar la ubicación cuando se aplique. Utilice la propiedad `x` de FLVPlayback y no la propiedad `MovieClip._x` porque la propiedad `_x` podría devolver un valor distinto si se carga un archivo SWF de aspecto.

Ejemplo

El siguiente ejemplo coloca la instancia de FLVPlayback a una distancia de 50 píxeles desde la izquierda y de 25 píxeles desde la parte superior.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.x = 50;
my_FLVPlybk.y = 25;
```

Véase también

`FLVPlayback.y`

FLVPlayback.y

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Sintaxis

my_FLVPlybk.y

Descripción

Propiedad; número que especifica la ubicación de la coordenada vertical del reproductor de vídeo. Esta propiedad sólo afecta a la ubicación vertical de la instancia de FLVPlayback y no incluye la ubicación de un archivo SWF de aspecto que pueda modificar la ubicación cuando se aplique. Utilice la propiedad `y` de FLVPlayback y no la propiedad `MovieClip._y` porque la propiedad `_y` podría devolver un valor distinto si se carga un archivo SWF de aspecto.

Ejemplo

El siguiente ejemplo coloca la instancia de FLVPlayback a una distancia de 25 píxeles desde la izquierda y de 50 píxeles desde la parte superior.

Arrastre un componente FLVPlayback al escenario y asígnele el nombre de instancia **my_FLVPlybk**. Añada el siguiente código al panel Acciones, en el fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente FLVPlayback en el escenario, con el nombre de instancia
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.x = 25;
my_FLVPlybk.y = 50;
```

Véase también

`FLVPlayback.x`

Clase VideoError

Herencia Error > VideoError

Nombre de clase de ActionScript `mx.video.VideoError`

Las propiedades de la clase VideoError permiten diagnosticar situaciones de error que se producen al trabajar con el componente FLVPlayback.

La clase mx.video.VideoError amplía la clase Error.

Resumen de propiedades de la clase VideoError

En la tabla siguiente se enumeran las propiedades de la clase VideoError:

Propiedad	Descripción
`VideoError.code`	Código de error numérico.
`VideoError.DELETE_DEFAULT_PLAYER`	Número que indica un intento de eliminar el reproductor de vídeo predeterminado.
`VideoError.DELETE_DEFAULT_PLAYER`	Número que indica un cuepoint no válido.
`VideoError.INVALID_CONTENT_PATH`	Número que indica un valor de `contentPath` no válido.
`VideoError.INVALID_SEEK`	Número que indica una búsqueda no válida.
`VideoError.INVALID_XML`	Número que indica que se ha encontrado XML no válido en un archivo XML.
`VideoError.NO_BITRATE_MATCH`	Número que indica que no se ha encontrado un archivo FLV predeterminado correspondiente a ninguna velocidad de transferencia.
`VideoError.NO_CONNECTION`	Número que indica que el método no se puede conectar al servidor o no encuentra el archivo FLV en el servidor.
`VideoError.NO_CUE_POINT_MATCH`	Número que indica que no se encontró ningún cuepoint coincidente.

VideoError.code

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.code`

Descripción

Código numérico que identifica la situación de error.

Ejemplo

El ejemplo siguiente muestra la situación de error en el panel Salida:

```
import mx.video.*;

try {
  ...
} catch (err:VideoError) {
  trace ("Error code is: " + err.code)
  ...
}
```

VideoError.DELETE_DEFAULT_PLAYER

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.DELETE_DEFAULT_PLAYER`

Descripción

Valor 1007, que se produce si se llama al método `FLVPlayback.closeVideoPlayer()` para intentar cerrar el reproductor de vídeo predeterminado (número 0). No se puede eliminar el reproductor de vídeo predeterminado.

Ejemplo

El siguiente código comprueba si se produce el código de error `DELETE_DEFAULT_PLAYER`:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == DELETE_DEFAULT_PLAYER) {
     ...
   }
}
```

Véase también

`FLVPlayback.activeVideoPlayerIndex`

VideoError.ILLEGAL_CUE_POINT

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

```
mx.video.VideoError.ILLEGAL_CUE_POINT
```

Descripción

Valor 1002, que indica que se encontró un cuepoint no válido.

Ejemplo

El siguiente código comprueba si se produce el código de error `ILLEGAL_CUE_POINT`:

```
try {
   ...
} catch (err:VideoError) {
   if (err.code == ILLEGAL_CUE_POINT) {
     ...
   }
}
```

Véase también

`FLVPlayback.cuePoint`

VideoError.INVALID_CONTENT_PATH

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.INVALID_CONTENT_PATH`

Descripción

Valor 1004, que indica que se encontró un valor de `contentPath` no válido.

Ejemplo

El siguiente código comprueba si se produce el código de error `INVALID_CONTENT_PATH`:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_CONTENT_PATH) {
     ...
   }
}
```

Véase también

`FLVPlayback.contentPath`

VideoError.INVALID_SEEK

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.INVALID_SEEK`

Descripción

Valor 1003, que indica que se intentó una búsqueda no válida.

Ejemplo

El siguiente código comprueba si se produce el código de error `INVALID_SEEK`:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_SEEK) {
     ...
   }
}
```

Véase también

`FLVPlayback.seek()`

VideoError.INVALID_XML

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.INVALID_XML`

Descripción

Valor 1005, que indica que se encontró XML no válido. Un error de XML no válido puede producirse cuando se descarga y analiza un archivo SMIL. La propiedad `VideoError.message` contiene texto que describe el problema concreto. Para más información, consulte "Utilización de un archivo SMIL" en la página 735.

Ejemplo

El siguiente código comprueba si se produce el código de error `INVALID_XML`:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_XML) {
     ...
   }
}
```

Véase también

`FLVPlayback.contentPath`

VideoError.NO_BITRATE_MATCH

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

`mx.video.VideoError.NO_BITRATE_MATCH`

Descripción

Valor 1006, que indica que no hay ningún archivo FLV predeterminado correspondiente a ninguna velocidad de transferencia. Sólo se produce cuando se descarga y analiza un archivo SMIL. Para más información, consulte "Utilización de un archivo SMIL" en la página 735.

Ejemplo

El siguiente código comprueba si se produce el código de error `NO_BITRATE_MATCH`:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_BITRATE_MATCH) {
     ...
   }
}
```

Véase también

FLVPlayback.bitrate

VideoError.NO_CONNECTION

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

mx.video.VideoError.NO_CONNECTION

Descripción

Valor 1000, que indica que el método no se puede conectar al servidor o no encuentra el archivo FLV en el servidor.

Ejemplo

El siguiente código comprueba si se produce el código de error NO_CONNECTION:

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_CONNECTION) {
     ...
   }
}
```

VideoError.NO_CUE_POINT_MATCH

Disponibilidad

Flash Player 8.

Edición

Flash Professional 8.

Utilización

mx.video.VideoError.NO_CUE_POINT_MATCH

Descripción

Valor 1001, que indica que no se encontró ningún cuepoint coincidente.

Ejemplo

El siguiente código comprueba si se produce el código de error `NO_CUE_POINT_MATCH` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_CUE_POINT_MATCH) {
     ...
   }
}
```

Véase también

`FLVPlayback.findCuePoint()`

Clase VideoPlayer

Herencia MovieClip > Clase VideoPlayer

Nombre de clase de ActionScript `mx.video.VideoPlayer`

VideoPlayer amplía la clase MovieClip y ajusta un objeto VideoPlayer.

La clase FLVPlayback ajusta la clase VideoPlayer y Macromedia le recomienda que la utilice en casi todos los casos. No hay ninguna función de la clase VideoPlayer a la que no pueda accederse mediante la clase FLVPlayback.

La clase VideoPlayer se incluye aquí porque permite crear un reproductor de vídeo con un archivo SWF de menor tamaño. La clase VideoPlayer no permite incluir un aspecto ni controles de reproducción y tiene una API más pequeña. Por ejemplo, no es posible realizar búsquedas de cuepoints, aunque sí se producirán eventos `cuePoint`.

Además, la clase FLVPlayback interactúa automáticamente con la clase NCManager para acceder a archivos FLV transmitidos desde un servidor FCS, por ejemplo. La interacción con la clase NCManager se produce al establecer la propiedad `contentPath` y al pasar una URL a los métodos `play()` y `load()`. Sin embargo, si utiliza la clase VideoPlayer de forma independiente, debe incluir la siguiente sentencia en el código ActionScript para asegurarse de que se incluye la clase NCManager:

```
_forceNCManager:mx.video.NCManager;
```

La clase NCManager también tiene una clase de interfaz, INCManager, que permite reemplazar la clase NCManager por una clase personalizada para administrar comunicaciones de red. Si lo hace, también necesitará incluir la siguiente sentencia para reemplazar `NCManager` por el nombre de la clase que haya proporcionado:

```
mx.video.VideoPlayer.DEFAULT_INCMANAGER = "mx.video.NCManager";
```

No es necesario añadir esta sentencia si utiliza la clase NCManager predeterminada.

NOTA	También puede establecer DEFAULT_INCMANAGER para reemplazar la clase mx.video.NCManager predeterminada por el componente FLVPlayback.

Para gestionar varios flujos en varios anchos de banda, NCManager admite un subconjunto de SMIL. Para más información, consulte "Utilización de un archivo SMIL" en la página 735.

En esta sección se ofrece un resumen de la clase VideoPlayer. Encontrará documentación detallada sobre los métodos, propiedades y eventos de la clase VideoPlayer en www.macromedia.com/go/videoplayer.

Resumen de métodos de la clase VideoPlayer

En la tabla siguiente se enumeran los métodos de la clase VideoPlayer:

Método	Descripción
`VideoPlayer.addEventListener()`	Crea un detector de un evento especificado.
`VideoPlayer.close()`	Cierra el flujo de vídeo y la conexión al servidor FCS.
`VideoPlayer.load()`	Carga el archivo FLV pero no inicia su reproducción. Después de cambiar el tamaño (si es necesario) se pausa el archivo FLV.
`VideoPlayer.pause()`	Pausa la reproducción del flujo de vídeo.
`VideoPlayer.play()`	Inicia la reproducción del flujo de vídeo.
`VideoPlayer.removeEventListener()`	Elimina un detector de eventos.
`VideoPlayer.seek()`	Busca un tiempo especificado en el archivo (en segundos, con una precisión decimal de milisegundos).
`VideoPlayer.setScale()`	Establece simultáneamente los valores de `scaleX` y `scaleY`.
`VideoPlayer.setSize()`	Establece simultáneamente los valores de `width` y `height`.
`VideoPlayer.stop()`	Detiene la reproducción del flujo de vídeo.

Resumen de propiedades de la clase VideoPlayer

La clase VideoPlayer tiene propiedades tanto de clase como de instancia.

Propiedades de clase

Las siguientes propiedades sólo se aplican a la clase VideoPlayer. Son constantes de sólo lectura que se aplican a todas las instancias de la clase VideoPlayer.

Propiedad	Valor	Descripción
`VideoPlayer.BUFFERING`	`"buffering"`	Valor posible para la propiedad `state`. Indica el estado que se adquiere inmediatamente después de llamar a `play()` o `load()`.
`VideoPlayer.CONNECTION_ERROR`	`"connectionError"`	Valor posible para la propiedad `state`. Indica que se ha producido un error de conexión.
`VideoPlayer.DEFAULT_INCMANAGER`	`“mx.video.NCManager”`	Nombre de la implementación predeterminada (mx.video.NCManager) o personalizada de la interfaz de INCManager.
`VideoPlayer.DISCONNECTED`	`"disconnected"`	Valor posible para la propiedad `state`. Indica que el flujo del archivo FLV está desconectado.
`VideoPlayer.LOADING`	`"loading"`	Valor posible para la propiedad `state`. Indica que el archivo FLV se está cargando.
`VideoPlayer.PAUSED`	`"paused"`	Valor posible para la propiedad `state`. Indica que el archivo FLV está en pausa.
`VideoPlayer.PLAYING`	`"playing"`	Valor posible para la propiedad `state`. Indica que el archivo FLV se está reproduciendo.

Propiedad	Valor	Descripción
VideoPlayer.RESIZING	"resizing"	Valor posible para la propiedad state. Indica que se está cambiando el tamaño del archivo FLV.
VideoPlayer.REWINDING	"rewinding"	Valor posible para la propiedad state. Indica que el archivo FLV se está rebobinando.
VideoPlayer.SEEKING	"seeking"	Valor posible para la propiedad state. Indica que el archivo FLV está buscando.
VideoPlayer.STOPPED	"stopped"	Valor posible para la propiedad state. Indica que el archivo FLV está detenido.
VideoPlayer.version	x.x.x.xx	Número de versión del componente.

Propiedades de instancia

En la tabla siguiente se enumeran las propiedades de instancia de la clase VideoPlayer. Este conjunto de propiedades se aplica a cada instancia de una clase VideoPlayer.

Propiedad	Descripción
VideoPlayer.autoRewind	Valor booleano. Si es true hace que se rebobine el archivo FLV hasta el primer fotograma cuando se detenga la reproducción.
VideoPlayer.autoSize	Valor booleano. Si es true, hace que se ajuste automáticamente el tamaño del vídeo a las dimensiones de origen.
VideoPlayer.bufferTime	Valor que especifica el número de segundos que se almacenarán en la memoria antes de que se inicie la reproducción de un flujo de vídeo.
VideoPlayer.bytesLoaded	Valor que indica el número de bytes descargados para una descarga HTTP. Sólo lectura.
VideoPlayer.bytesTotal	Valor que especifica el número total de bytes descargados para una descarga HTTP. Sólo lectura.
VideoPlayer.connected	Valor booleano que indica si el flujo de archivo FLV está conectado. Sólo lectura.

Propiedad	Descripción
`VideoPlayer.height`	Número que especifica la altura del vídeo en píxeles.
`VideoPlayer.idleTimeout`	Cantidad de tiempo en milisegundos antes de que se cierre una conexión inactiva al servidor FCS tras pausar o detener una reproducción.
`VideoPlayer.isLive`	Valor booleano. Es `true` si el flujo de vídeo es dinámico. No se aplica a las descargas HTTP.
`VideoPlayer.isRTMP`	Valor booleano. Es `true` si el archivo FLV se transmite desde un servidor FCS. Sólo lectura.
`VideoPlayer.maintainAspectRatio`	Valor booleano. Si es `true`, mantiene la proporción de vídeo.
`VideoPlayer.metadata`	Objeto que es un paquete de información de metadatos que se recibe de una llamada a la función callback `onMetaData()`, si está disponible. Sólo lectura.
`VideoPlayer.ncMgr`	Objeto INCManager que proporciona acceso a una instancia de la clase que implementa INCManager.
`VideoPlayer.playheadTime`	Número que representa el tiempo o la posición actual (en segundos) de la cabeza lectora, que puede ser un valor fraccionario.
`VideoPlayer.playheadUpdateInterval`	Número que es la cantidad de tiempo en milisegundos entre cada evento `playheadUpdate`.
`VideoPlayer.progressInterval`	Número que es la cantidad de tiempo en milisegundos entre cada evento progress.
`VideoPlayer.scaleX`	Número que especifica la escala horizontal.
`VideoPlayer.scaleY`	Número que especifica la escala vertical.
`VideoPlayer.state`	Cadena que especifica el estado del componente. Se establece con los métodos `load()`, `play()`, `stop()`, `pause()` y `seek()`. Sólo lectura.
`VideoPlayer.stateResponsive`	Valor booleano. Es `true` si el estado es interactivo (es decir, si se pueden activar los controles en el estado actual). Sólo lectura.
`VideoPlayer.totalTime`	Número que representa el tiempo total de reproducción del vídeo.
`VideoPlayer.transform`	Objeto que proporciona acceso directo a los métodos `Sound.setTransform()` y `Sound.getTransform()` para proporcionar mayor control del sonido.

Propiedad	Descripción
`VideoPlayer.url`	Cadena que especifica la URL del flujo cargado (o que se está cargando).
`VideoPlayer.videoHeight`	Número que especifica la altura del archivo FLV.
`VideoPlayer.videoWidth`	Número que especifica la anchura del archivo FLV.
`VideoPlayer.visible`	Valor booleano. Si es `true`, hace que el archivo FLV sea visible.
`VideoPlayer.volume`	Número del intervalo 0 a 100 que indica el nivel del control de volumen.
`VideoPlayer.width`	Número (porcentaje) que especifica hasta dónde puede mover un usuario el selector de la barra de volumen antes de que se produzca una actualización.
`VideoPlayer.x`	Número que especifica la dimensión horizontal en píxeles del reproductor de vídeo.
`VideoPlayer.y`	Número que especifica la dimensión vertical en píxeles del reproductor de vídeo.

Resumen de eventos de la clase VideoPlayer

En la tabla siguiente se enumeran los eventos de la clase VideoPlayer:

Evento	Descripción
`VideoPlayer.close`	Se distribuye cuando se cierra el flujo de vídeo porque se agotó el tiempo de espera o mediante una llamada al método `close()`.
`VideoPlayer.complete`	Se distribuye cuando finaliza la reproducción al llegar al final del archivo FLV.
`VideoPlayer.cuePoint`	Se distribuye cuando se llega a un cuepoint.
`VideoPlayer.metadataReceived`	Se distribuye la primera vez que se llega a los metadatos del archivo FLV.
`VideoPlayer.playheadUpdate`	Se distribuye cada 0,25 segundos mientras se está reproduciendo el archivo FLV.
`VideoPlayer.progress`	Se distribuye cada 0,25 segundos, desde que se llama al método `load()` hasta que se hayan cargado todos los bytes o se produzca un error de red.

Evento	Descripción
`VideoPlayer.ready`	Se distribuye cuando el archivo FLV está cargado y preparado para mostrarse.
`VideoPlayer.resize`	Se distribuye cuando se cambia el tamaño del vídeo.
`VideoPlayer.rewind`	Se distribuye cuando se mueve hacia atrás la ubicación de la cabeza lectora mediante una llamada a `seek()` o cuando finaliza la operación de rebobinado automático.
`VideoPlayer.stateChange`	Se distribuye cuando cambia el estado de reproducción.

Utilización de un archivo SMIL

Para gestionar varios flujos en varios anchos de banda, la clase VideoPlayer utiliza una clase auxiliar (NCManager) que admite un subconjunto de SMIL. SMIL se utiliza para identificar la ubicación del flujo de vídeo, el diseño (anchura y altura) del archivo FLV y los archivos FLV de origen correspondientes a los distintos anchos de banda. También se puede utilizar para especificar la velocidad de transferencia y la duración del archivo FLV.

El siguiente ejemplo muestra un archivo SMIL que transmite archivos FLV de distintos anchos de banda desde un servidor FCS, mediante RTMP:

```
<smil>
    <head>
        <meta base="rtmp://myserver/mypgm/" >
        <layout>
            <root-layout width="240" height="180" >
        </layout>
    </head>
    <body>
        <switch>
            <video src="myvideo_mdm.flv" system-bitrate="56000"
  dur="3:00.1">
            <video src="myvideo_isdn.flv" system-bitrate="128000"
  dur="3:00.1">
            <ref src="myvideo_cable.flv" dur="3:00.1"/>
        </switch>
    </body>
</smil>
```

La etiqueta `<head>` puede contener las etiquetas `<meta>` y `<layout>`. La etiqueta `<meta>` sólo admite el atributo `base`, que se utiliza para especificar la URL del vídeo de flujo (RTMP desde un servidor FCS).

La etiqueta `<layout>` sólo admite el elemento `root-layout`, que se utiliza para establecer los atributos `height` y `width`; por lo tanto, determina el tamaño de la ventana en la que se representa el archivo FLV. Estos atributos sólo aceptan valores de píxeles, no porcentajes.

En el cuerpo del archivo SMIL, puede incluir un único vínculo a un archivo FLV de origen o, si transmite varios archivos para distintos anchos de banda desde un servidor FCS (como en el ejemplo anterior), puede utilizar la etiqueta `<switch>` para enumerar los archivos de origen.

Las etiquetas `video` y `ref` contenidas en la etiqueta `<switch>` son sinónimas: ambas pueden utilizar el atributo `src` para especificar archivos FLV. Además, cada una de ellas puede utilizar los atributos `region`, `system-bitrate` y `dur` para especificar la región, el ancho de banda mínimo requerido y la duración del archivo FLV.

En la etiqueta `<body>`, sólo se permite que las etiquetas `<video>`, `<src>` o `<switch>` aparezcan una vez.

El siguiente ejemplo muestra una descarga progresiva de un solo archivo FLV que no utiliza la detección de ancho de banda:

```
<smil>
     <head>
       <layout>
         <root-layout width="240" height="180" />
       </layout>
     </head>
     <body>
       <video src=""myvideo.flv" />
     </body>
</smil>
```

<smil>

Disponibilidad

Flash Professional 8.

Utilización

```
<smil>
...
child tags
...
</smil>
```

Attributes

Ninguno.

Etiquetas secundarias

`<head>`, `<body>`

Etiqueta principal

Ninguna.

Descripción

Etiqueta de nivel superior, que identifica un archivo SMIL.

Ejemplo

El siguiente ejemplo muestra un archivo SMIL que especifica tres archivos FLV:

```
<smil>
    <head>
        <meta base="rtmp://myserver/mypgm/" >
        <layout>
            <root-layout width="240" height="180" >
        </layout>
    </head>
    <body>
        <switch>
            <video src="myvideo_mdm.flv" system-bitrate="56000"
  dur="3:00.1">
            <video src="myvideo_isdn.flv" system-bitrate="128000"
  dur="3:00.1">
            <ref src="myvideo_cable.flv" dur="3:00.1"/>
        </switch>
    </body>
</smil>
```

<head>

Disponibilidad

Flash Professional 8.

Utilización

```
<head>
...
child tags
...
</head>
```

Attributes

Ninguno.

Etiquetas secundarias

`<meta>`, `<layout>`

Etiqueta principal

`<smil>`

Descripción

Admite las etiquetas `<meta>` y `<layout>` y especifica la ubicación y el diseño predeterminado (altura y anchura) de los archivos FLV de origen.

Ejemplo

```
El siguiente ejemplo establece el diseño raíz en 240 píxeles por 180
  píxeles:
<head>
  <meta base="rtmp://myserver/mypgm/" >
  <layout>
    <root-layout width="240" height="180" >
  </layout>
</head>
```

<meta>

Disponibilidad

Flash Professional 8.

Utilización

`<meta/>`

Attributes

`base`

Etiquetas secundarias

`<layout>`

Etiqueta principal

Ninguno.

Descripción

Contiene el atributo `base`, que especifica la ubicación (URL RTMP) de los archivos FLV de origen.

Ejemplo

El siguiente ejemplo muestra una etiqueta meta de una ubicación de base en `myserver`:

```
<meta base="rtmp://myserver/mypgm/" >
```

<layout>

Disponibilidad

Flash Professional 8.

Utilización

```
<layout>
...
child tags
...
</layout>
```

Attributes

Ninguno.

Etiquetas secundarias

```
<root-layout>
```

Etiqueta principal

```
<meta>
```

Descripción

Especifica la anchura y la altura del archivo FLV.

Ejemplo

El siguiente ejemplo establece el diseño en 240 píxeles por 180 píxeles:

```
<layout>
  <root-layout width="240" height="180" >
</layout>
```

<root-layout>

Disponibilidad

Flash Professional 8.

Utilización

```
<root-layout...attributes.../>
```

Attributes

Anchura, altura

Etiquetas secundarias

`None.`

Etiqueta principal

`<layout>`

Descripción

Especifica la anchura y la altura del archivo FLV.

Ejemplo

El siguiente ejemplo establece el diseño en 240 píxeles por 180 píxeles:

```
<root-layout width="240" height="180" >
```

‹body›

Disponibilidad

Flash Professional 8.

Utilización

```
<body>
...
child tags
...
</body>
```

Attributes

Ninguno.

Etiquetas secundarias

`<video>`, `<ref>`, `<switch>`

Etiqueta principal

`<smil>`

Descripción

Contiene las etiquetas `<video>`, `<ref>` y `<switch>`, que especifican el nombre del archivo FLV de origen, el ancho de banda mínimo y la duración del archivo FLV. El atributo `system-bitrate` sólo se admite cuando se utiliza la etiqueta `<switch>`. En la etiqueta `<body>`, sólo se permite que las etiquetas `<switch>`, `<video>` o `<ref>` aparezcan una vez.

Ejemplo

El siguiente ejemplo especifica tres archivos FLV, dos de ellos con la etiqueta `video` y el otro con la etiqueta `ref`:

```
<body>
  <switch>
    <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
    <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1">
    <ref src="myvideo_cable.flv" dur="3:00.1"/>
  </switch>
</body>
```

<video>

Disponibilidad

Flash Professional 8.

Utilización

`<video...attributes.../>`

Attributes

`src`, `system-bitrate`, `dur`

Etiquetas secundarias

`None`

Etiqueta principal

`<body>`

Descripción

Sinónimo de la etiqueta `<ref>`. Admite los atributos `src` y `dur`, que especifican el nombre del archivo FLV de origen y su duración. El atributo `dur` admite los formatos de tiempo completo (00:03:00:01) y parcial (03:00:01).

Ejemplo

El siguiente ejemplo establece el origen y la duración de un vídeo:

```
<video src="myvideo_mdm.flv" dur="3:00.1">
```

<ref>

Disponibilidad

Flash Professional 8.

Utilización

```
<ref...attributes.../>
```

Attributes

`src`, `system-bitrate`, `dur`

Etiquetas secundarias

`None`

Etiqueta principal

`<body>`

Descripción

Sinónimo de la etiqueta `<video>`. Admite los atributos `src` y `dur`, que especifican el nombre del archivo FLV de origen y su duración. El atributo `dur` admite los formatos de tiempo completo (00:03:00:01) y parcial (03:00:01).

Ejemplo

El siguiente ejemplo establece el origen y la duración de un vídeo:

```
<ref src="myvideo_cable.flv" dur="3:00.1"/>
```

<switch>

Disponibilidad

Flash Professional 8.

Utilización

```
<switch>
...
child tags
...
<switch/>
```

Attributes

Ninguno

Etiquetas secundarias

`<video>`, `<ref>`

Etiqueta principal

`<body>`

Descripción

Se utiliza con las propiedades secundarias `<video>` o `<ref>` para enumerar los archivos FLV de flujo de vídeo transmitido en varios anchos de banda. La etiqueta `<switch>` admite el atributo `system-bitrate`, que especifica el ancho de banda mínimo, y los atributos `src` y `dur`.

Ejemplo

El siguiente ejemplo especifica tres archivos FLV, dos de ellos con la etiqueta `video` y el otro con la etiqueta `ref`:

```
<switch>
  <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
  <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1">
  <ref src="myvideo_cable.flv" dur="3:00.1"/>
</switch>
```

CAPÍTULO 23

Clase FocusManager

23

Se puede utilizar la clase Focus Manager para especificar el orden en el que se seleccionan los componentes cuando el usuario presiona el tabulador para desplazarse por una aplicación. También es posible emplear Focus Manager para definir un botón en el documento que reciba la entrada del teclado cuando el usuario presione Intro (Windows) o Retorno (Macintosh). Por ejemplo, cuando los usuarios rellenan un formulario, deben pasar de un campo a otro con el tabulador y presionar Intro (Windows) o Retorno (Macintosh) para enviar el formulario.

Todos los componentes implementan compatibilidad con Focus Manager; no es necesario escribir código alguno para invocar la clase FocusManager.

NOTA

La compatibilidad con Focus Manager sustituye el uso del controlador global `on(keyPress)`. Dado que todos los componentes implementan Focus Manager, una aplicación que incluya componentes *y* utilice el controlador global `on(keyPress)`, deberá tener establecido de forma explícita un `tabIndex` para cada control (incluidos los componentes *y* clips de película). Véase "Utilización de Focus Manager" en la página 746. O bien, es preferible añadir un detector de eventos para cada tecla específica y Focus Manager no sustituirá al controlador de eventos correspondiente. Para más información sobre la creación de un detector de eventos de una tecla, consulte "Captura de teclas presionadas" en *Aprendizaje de ActionScript 2.0 en Flash*.

Focus Manager interactúa con System Manager, que activa y desactiva instancias de FocusManager a medida que se activan y desactivan ventanas emergentes. Cada ventana modal tiene una instancia de FocusManager, por lo que los componentes de la ventana se convierten en su propio conjunto de tabuladores, lo que impide que el usuario pase a componentes de otras ventanas al tabular.

Focus Manager reconoce los grupos de botones de opción (los que tienen definida una propiedad `RadioButton.groupName`) y selecciona la instancia del grupo que tiene una propiedad `selected` definida en `true`. Cuando se presiona el tabulador, Focus Manager comprueba si el objeto siguiente tiene el mismo nombre de grupo que el objeto activo. Si es así, desplaza automáticamente la selección al objeto siguiente que tenga un nombre de grupo diferente. También pueden utilizar esta función los grupos de componentes que admitan la propiedad `groupName`.

Focus Manager controla los cambios de selección producidos con clics del ratón. Si el usuario hace clic en un componente, el componente se selecciona.

NOTA	Para probar un script con Focus Manager (Control › Probar película), seleccione Control › Deshabilitar métodos abreviados de teclado en modo Probar película; de lo contrario, Focus Manager no parecerá que funcione. Además, la tabulación y los métodos abreviados del teclado se utilizan en el entorno de edición de forma predeterminada. Así pues, si utiliza el modo de prueba, la navegación mediante tabuladores, la tecla Intro y otras combinaciones de teclas, pueden producirse resultados inesperados. Estas funciones deberían probarse en Flash Player fuera del entorno de edición.

Utilización de Focus Manager

Focus Manager no selecciona automáticamente ningún componente. Debe escribir un script que efectúe una llamada a `FocusManager.setFocus()` para un componente si desea que dicho componente esté seleccionado cuando se cargue una aplicación.

NOTA	Si efectúa una llamada a `FocusManager.setFocus()` para que un componente esté seleccionado cuando se cargue una aplicación, el anillo de selección no aparece alrededor de dicho componente. El componente está seleccionado, pero el indicador no está presente.

Para que la selección pueda desplazarse por la aplicación, defina la propiedad `tabIndex` en los objetos que deban seleccionarse (botones incluidos). Cuando el usuario presione el tabulador, Focus Manager buscará el objeto activado cuya propiedad `tabIndex` sea superior al valor actual de `tabIndex`. Cuando el administrador de selección alcance la propiedad `tabIndex` más alta, volverá a cero. De este modo, en el ejemplo siguiente el objeto `comment` (probablemente un componente TextArea) es el primero en seleccionarse y después se selecciona el objeto `okButton`:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

También se puede utilizar el panel Accesibilidad para asignar un valor de índice de tabulación.

Si ningún elemento del escenario tiene un valor de índice de tabulación, Focus Manager utiliza la *profundidad* (orden de apilamiento u orden *z*). La profundidad se configura principalmente con el orden en el que los componentes se arrastran al escenario. Sin embargo, también se pueden utilizar los comandos Modificar > Organizar > Traer al frente/Enviar al fondo para determinar la profundidad final.

Para crear un botón que se seleccione cuando el usuario presione la tecla Intro (Windows) o Retorno (Macintosh), defina la propiedad `FocusManager.defaultPushButton` en el nombre de instancia del botón deseado, como se muestra a continuación:

```
focusManager.defaultPushButton = okButton;
```

NOTA Focus Manager detecta cuándo se colocan los objetos en el escenario (el orden de profundidad de los objetos), pero no sus posiciones relativas en el mismo. Es diferente al modo en el que Flash Player controla la tabulación.

Utilización de Focus Manager para permitir la tabulación

Se puede utilizar Focus Manager para crear un esquema que permita a los usuarios presionar el tabulador y desplazarse por los objetos de una aplicación Flash. Los objetos del esquema de tabulación se denominan *destinos de tabulación*. Focus Manager examina las propiedades `tabEnabled` y `tabChildren` de los elementos principales de los objetos a fin de localizar los objetos.

Un clip de película puede ser un contenedor de destinos de tabulación, un destino de tabulación o ninguno de éstos:

Tipo de clip de película	tabEnabled	tabChildren
Contenedor de destinos de tabulación	`false`	`true`
Destino de tabulación	`true`	`false`
Ninguno	`false`	`false`

NOTA Esto resulta diferente al comportamiento predeterminado de Flash Player, en el que la propiedad `tabChildren` de un contenedor puede ser `undefined`.

Observe el siguiente caso: En el escenario de la línea de tiempo principal, hay dos campos de texto (`txt1` y `txt2`), así como un clip de película (`mc`) que incluye un componente DataGrid (`grid1`) y otro campo de texto (`txt3`). Se usará el siguiente código para permitir que los usuarios presionen el tabulador y se desplacen por los objetos en el orden indicado a continuación: `txt1`, `txt2`, `grid1`, `txt3`.

NOTA Las instancias de FocusManager y TextField están activadas de forma predeterminada.

```
// Notificar a Focus Manager que mc tiene elementos secundarios;
// esto sustituye mc.focusEnabled=true;
mc.tabChildren=true;
mc.tabEnabled=false;
// Establecer secuencia de tabulación.
txt1.tabIndex = 1;
txt2.tabIndex = 2;
mc.grid1.tabIndex = 3;
mc.txt3.tabIndex = 4;

// Establecer txt 1 como selección inicial.
txt1.text = "focus";
focusManager.setFocus(txt1);
```

Si el clip de película no tiene un método `onPress` o `onRelease`, o bien una propiedad `tabEnabled`, Focus Manager no lo verá a menos que establezca `focusEnabled` en `true`. Los campos de introducción de texto siempre se encuentran en el esquema de tabulación a menos que estén desactivados.

Si se está reproduciendo una aplicación Flash en un navegador Web, la aplicación no estará seleccionada hasta que un usuario haga clic en algún lugar de la aplicación. Asimismo, una vez que un usuario haga clic en la aplicación Flash, es posible que la selección salte fuera de dicha aplicación al presionar el tabulador. Para mantener la tabulación limitada a los objetos dentro de la aplicación Flash en el control ActiveX de Flash Player 7, añada el siguiente parámetro a la etiqueta HTML `<object>`:

```
<param name="SeamlessTabbing" value="false"/>
```

Creación de aplicaciones con Focus Manager

El siguiente procedimiento crea un esquema de selección en una aplicación Flash.

Para crear un esquema de selección:

1. Arrastre el componente TextInput desde el panel Componentes al escenario.
2. En el inspector de propiedades, asígnele el nombre de instancia **comment**.
3. Arrastre el componente Button desde el panel Componentes al escenario.
4. En el inspector de propiedades, asígnele el nombre de instancia **okButton** y defina el parámetro de etiqueta en **OK**.

5. En el fotograma 1 del panel Acciones, introduzca lo siguiente:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
focusManager.setFocus(comment);
function click(evt){
  trace(evt.type);
}
okButton.addEventListener("click", this);
```

6. Seleccione Control > Probar película.
7. Seleccione Control > Deshabilitar métodos abreviados de teclado.

   El código establece el orden de tabulación. Aunque el campo de comentarios no tiene un anillo de selección inicial, sí que se selecciona de forma inicial; puede empezar a escribir en dicho campo sin necesidad de hacer clic en él. Además, tiene que seleccionar la opción de menú Deshabilitar métodos abreviados de teclado para que funcione correctamente en el modo de prueba.

Personalización de Focus Manager

Se puede cambiar el color del anillo de selección en el tema Halo cambiando el valor del estilo `themeColor`, como en este ejemplo:

```
_global.style.setStyle("themeColor", "haloBlue");
```

Focus Manager utiliza un aspecto FocusRect para seleccionar. Este aspecto puede reemplazarse o modificarse, y es posible sustituir `UIComponent.drawFocus` por subclases para emplear indicadores de selección personalizados.

Clase FocusManager (API)

Herencia MovieClip > Clase UIObject > Clase UIComponent > FocusManager

Nombre de clase de ActionScript mx.managers.FocusManager

Se puede utilizar Focus Manager para especificar el orden en el que se seleccionan los componentes cuando el usuario presiona el tabulador para desplazarse por una aplicación. Asimismo, es posible emplear la clase FocusManager para definir un botón en el documento que reciba la entrada del teclado cuando el usuario presione Intro (Windows) o Retorno (Macintosh).

SUGERENCIA	En un archivo de clase que reciba una herencia de UIComponent, no se recomienda hacer referencia a `_root.focusManager`. Cada instancia de UIComponent hereda un método `getFocusManager()`, que devuelve una referencia a la instancia de FocusManager responsable de controlar el esquema de selección del componente.

Resumen de métodos de la clase FocusManager

En la tabla siguiente se enumeran los métodos de la clase FocusManager.

Método	Descripción
`FocusManager.getFocus()`	Devuelve una referencia al objeto seleccionado.
`FocusManager.sendDefaultPushButtonEvent()`	Envía un evento `click` a los objetos detectores registrados en el botón de comando predeterminado.
`FocusManager.setFocus()`	Selecciona el objeto especificado.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase FocusManager de la clase UIObject.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.

Método	Descripción
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase FocusManager de la clase UIComponent.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase FocusManager

En la tabla siguiente se enumeran las propiedades de la clase FocusManager.

Propiedad	Descripción
FocusManager.defaultPushButton	Objeto que recibe el evento `click` cuando el usuario presiona la tecla Intro (Windows) o Retorno (Macintosh).
FocusManager.defaultPushButtonEnabled	Indica si la gestión con el teclado está activada (`true`) o desactivada (`false`) para el botón de comando predeterminado. El valor predeterminado es `true`.
FocusManager.enabled	Indica si la gestión con el tabulador está activada (`true`) o desactivada (`false`). El valor predeterminado es `true`.
FocusManager.nextTabIndex	Valor siguiente de la propiedad `tabIndex`.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase FocusManager de la clase UIObject.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase FocusManager de la clase UIComponent.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase FocusManager

No hay eventos exclusivos de la clase FocusManager.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase FocusManager de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase FocusManager de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

FocusManager.defaultPushButton

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
focusManager.defaultPushButton
```

Descripción

Propiedad; especifica el botón de comando predeterminado para una aplicación. Cuando el usuario presiona Intro (Windows) o Retorno (Macintosh), los detectores del botón de comando predeterminado reciben un evento `click`. El valor predeterminado es `undefined` y el tipo de datos de esta propiedad es objeto.

Focus Manager utiliza la declaración de estilo emphasized de la clase SimpleButton para indicar visualmente el botón de comando predeterminado actual.

El valor de la propiedad `defaultPushButton` es siempre el botón seleccionado. El botón de comando predeterminado no se selecciona inicialmente por haber definido la propiedad `defaultPushButton`. Si hay varios botones en una aplicación, el botón seleccionado actualmente recibe el evento `click` cuando se presiona la tecla Intro o Retorno. Si hay algún otro componente seleccionado cuando se presiona Intro o Retorno, se restablece el valor original de la propiedad `defaultPushButton`.

Ejemplo

El código siguiente define el botón de comando predeterminado en la instancia `OKButton`:

```
focusManager.defaultPushButton = OKButton;
```

Véase también

`FocusManager.defaultPushButtonEnabled`,
`FocusManager.sendDefaultPushButtonEvent()`

FocusManager.defaultPushButtonEnabled

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
focusManager.defaultPushButtonEnabled
```

Descripción

Propiedad; valor booleano que determina si la gestión con el teclado está activada (`true`) o no (`false`) para el botón de comando predeterminado. Si `defaultPushButtonEnabled` se define con el valor `false`, los componentes pueden recibir la entrada de las teclas Retorno o Intro y gestionarla internamente. Deberá volver a activar la gestión con el botón de comando predeterminado observando el método `onKillFocus()` del componente (véase %{onKillFocus (MovieClip.onKillFocus handler)}% en *Referencia del lenguaje ActionScript 2.0*) o el evento `focusOut`. El valor predeterminado es `true`.

Los desarrolladores de componentes avanzados utilizan esta propiedad.

Ejemplo

El código siguiente desactiva la gestión con el botón de comando predeterminado:

```
focusManager.defaultPushButtonEnabled = false;
```

FocusManager.enabled

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
focusManager.enabled
```

Descripción

Propiedad; valor booleano que determina si una gestión con el tabulador está activada (`true`) o no (`false`) para un grupo determinado de objetos de selección. Por ejemplo, otra ventana emergente podría tener su propio Focus Manager. Si `enabled` se define en `false`, los componentes pueden recibir la entrada del tabulador y gestionarla internamente. Deberá volver a activar la gestión con Focus Manager observando el método `onKillFocus()` del componente (véase %{onKillFocus (MovieClip.onKillFocus handler)}% en *Referencia del lenguaje ActionScript 2.0*) o el evento `focusOut`. El valor predeterminado es `true`.

Ejemplo

El código siguiente desactiva la tabulación:

```
focusManager.enabled = false;
```

FocusManager.getFocus()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
focusManager.getFocus()
```

Parámetros

Ninguno.

Valor devuelto

Referencia al objeto seleccionado.

Descripción

Método; devuelve una referencia al objeto seleccionado actualmente.

Ejemplo

El código siguiente selecciona `myOKButton` si el objeto actualmente seleccionado es `myInputText`:

```
if (focusManager.getFocus() == myInputText)
{
  focusManager.setFocus(myOKButton);
}
```

Véase también

FocusManager.setFocus()

FocusManager.nextTabIndex

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
FocusManager.nextTabIndex
```

Descripción

Propiedad; siguiente número de índice de tabulador disponible. Utilice esta propiedad para definir dinámicamente la propiedad `tabIndex` de un objeto.

Ejemplo

El código siguiente asigna a la instancia `mycheckbox` el siguiente valor `tabIndex` más alto:

```
mycheckbox.tabIndex = focusManager.nextTabIndex;
```

Véase también

UIComponent.tabIndex

FocusManager.sendDefaultPushButtonEvent()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
focusManager.sendDefaultPushButtonEvent()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; envía un evento `click` a los objetos detectores registrados en el botón de comando predeterminado. Utilice este método para enviar un evento `click` mediante programación.

Ejemplo

El código siguiente activa el evento `click` del botón de comando predeterminado y rellena los campos de nombre de usuario y contraseña cuando un usuario selecciona la instancia de CheckBox `chb` (la casilla de verificación debe tener la etiqueta "Conexión automática"):

```
name_txt.tabIndex = 1;
password_txt.tabIndex = 2;
chb.tabIndex = 3;
submit_ib.tabIndex = 4;

focusManager.defaultPushButton = submit_ib;

chbObj = new Object();
chbObj.click = function(o){
  if (chb.selected == true){
    name_txt.text = "Jody";
    password_txt.text = "foobar";
    focusManager.sendDefaultPushButtonEvent();
  } else {
    name_txt.text = "";
    password_txt.text = "";
  }
}
chb.addEventListener("click", chbObj);

submitObj = new Object();
submitObj.click = function(o){
  if (password_txt.text != "foobar"){
    trace("error on submit");
  } else {
    trace("Yeah! sendDefaultPushButtonEvent worked!");
  }
}
submit_ib.addEventListener("click", submitObj);
```

Véase también

`FocusManager.defaultPushButton`

FocusManager.setFocus()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
focusManager.setFocus(object)
```

Parámetros

object Referencia al objeto que debe seleccionarse.

Valor devuelto

Ninguno.

Descripción

Método; selecciona el objeto especificado. Si el objeto que desea seleccionar no se encuentra en la línea de tiempo principal, utilice el código siguiente:

```
_root.focusManager.setFocus(object);
```

Ejemplo

El código siguiente selecciona `myOKButton`:

```
focusManager.setFocus(myOKButton);
```

Véase también

`FocusManager.getFocus()`

CAPÍTULO 24

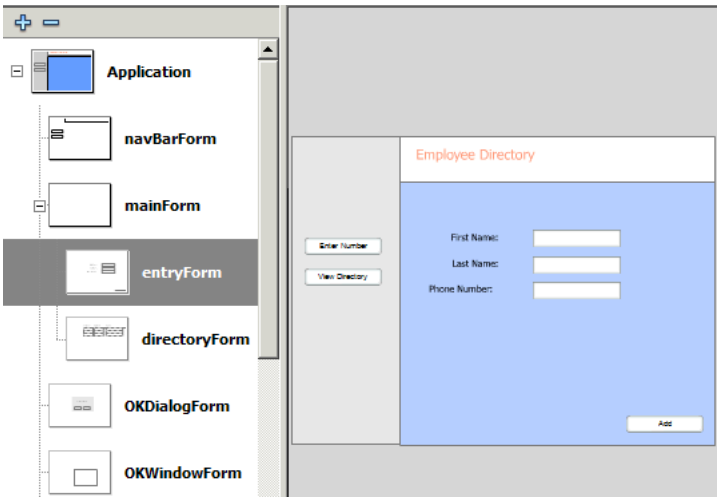
Clase Form (sólo en Flash Professional)

24

La clase Form proporciona el comportamiento en tiempo de ejecución de los formularios creados por el usuario en el panel Contorno de pantalla en Flash. Para ver información general sobre cómo trabajar con pantallas, consulte “Trabajo con pantallas (sólo para Flash Professional)” en *Utilización de Flash*.

Utilización de la clase Form (sólo en Flash Professional)

Los formularios desempeñan la función de contenedores de objetos gráficos (elementos de interfaz de usuario de una aplicación, por ejemplo) y de estados de aplicación. En el panel Contorno de pantalla puede visualizar los diferentes estados de la aplicación que está creando, donde cada formulario es un estado de aplicación diferente. Por ejemplo, en la ilustración siguiente se muestra el panel Contorno de pantalla de una aplicación de ejemplo que se ha diseñado con formularios.



Vista de contorno de pantalla de una aplicación de formulario de ejemplo

En la ilustración se muestra el contorno de una aplicación de ejemplo de directorio de empleados denominada Employee Directory, compuesta por varios formularios. El formulario denominado entryForm (que aparece seleccionado en la ilustración anterior) contiene varios objetos de interfaz de usuario, entre ellos campos de introducción de texto, etiquetas y un botón de comando. El desarrollador puede presentar fácilmente este formulario al usuario alternando su visibilidad (mediante la propiedad `Form.visible`), a la vez que alterna la visibilidad de otros formularios.

En el panel Comportamientos también puede asociar comportamientos y controles con los formularios. Para más información sobre cómo añadir transiciones y controles a las pantallas, consulte “Creación de controles y transiciones para las pantallas con comportamientos (sólo en Flash Professional)” en *Utilización de Flash*.

Puesto que la clase Form es una extensión de la clase Loader, puede cargar fácilmente contenido externo (un archivo SWF o un archivo JPEG) en un formulario. Por ejemplo, el contenido de un formulario puede ser un archivo SWF independiente, que puede contener formularios. De este modo, se pueden dividir en módulos las aplicaciones de formularios, lo cual facilita el mantenimiento de las aplicaciones y reduce el tiempo de descarga inicial. Para más información, consulte "Carga de contenido externo en pantallas (sólo en Flash Professional)" en la página 1110.

Parámetros de Form

A continuación se indican los parámetros de edición que se pueden definir para cada instancia de Form en el inspector de propiedades o en el inspector de componentes:

autoload indica si el contenido especificado por el parámetro contentPath debe cargarse automáticamente (`true`) o si debe esperar hasta que se llame al método `Loader.load()` (`false`). El valor predeterminado es `true`.

contentPath especifica el contenido del formulario. Este valor puede ser el identificador de vinculación de un clip de película o la URL absoluta o relativa de un archivo SWF o JPEG que se carga en la diapositiva. De forma predeterminada, el contenido cargado se recorta para que quepa en la diapositiva.

visible especifica si el formulario se ve (`true`) o está oculto (`false`) al cargarse la primera vez.

Clase Form (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > Componente Loader > Clase Screen (sólo en Flash Professional) > Form

Nombre de clase de ActionScript mx.screens.Form

La clase Form proporciona el comportamiento en tiempo de ejecución de los formularios creados por el usuario en el panel Contorno de pantalla en Flash.

Resumen de métodos de la clase Form

En la tabla siguiente se enumeran los métodos de la clase Form.

Método	Descripción
`Form.getChildForm()`	Devuelve el formulario secundario en un índice especificado.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Form de la clase UIObject. Al llamar a estos métodos desde el objeto Form, debe utilizarse la sintaxis *formInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Form de la clase UIComponent. Al llamar a estos métodos desde el objeto Form, debe utilizarse la sintaxis *formInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Métodos heredados de la clase Loader

En la tabla siguiente se enumeran los métodos que hereda la clase Form de la clase Loader. Al llamar a estos métodos desde el objeto Form, debe utilizarse la sintaxis *formInstance.methodName*.

Método	Descripción
Loader.load()	Carga el contenido especificado en la propiedad contentPath.

Métodos heredados de la clase Screen

En la tabla siguiente se enumeran los métodos que hereda la clase Form de la clase Screen. Al llamar a estos métodos desde el objeto Form, debe utilizarse la sintaxis *formInstance.methodName*.

Método	Descripción
Screen.getChildScreen()	Devuelve la pantalla secundaria de esta pantalla en un índice concreto.

Resumen de propiedades de la clase Form

En la tabla siguiente se enumeran las propiedades exclusivas de la clase Form.

Propiedad	Descripción
Form.currentFocusedForm	Sólo lectura; devuelve el formulario que contiene la selección actual global.
Form.indexInParentForm	Sólo lectura, devuelve el índice (basado en cero) de este formulario en la lista de subformularios de su formulario principal.
Form.numChildForms	Sólo lectura; devuelve el número de formularios secundarios que contiene este formulario.
Form.parentIsForm	Sólo lectura; especifica si el objeto principal de este formulario es también un formulario.
Form.parentForm	Sólo lectura; referencia al formulario principal del formulario.
Form.rootForm	Sólo lectura; devuelve la raíz del árbol de formularios, o subárbol, que contiene el formulario.
Form.visible	Especifica si el formulario está visible cuando se puede ver su formulario, diapositiva, clip de película o archivo SWF principal.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Form de la clase UIObject. Al acceder a estas propiedades desde el objeto Form, debe utilizarse la sintaxis *formInstance.propertyName*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Form de la clase UIComponent. Al acceder a estas propiedades desde el objeto Form, debe utilizarse la sintaxis *formInstance.propertyName*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase Loader

En la tabla siguiente se enumeran las propiedades que hereda la clase Form de la clase Loader. Al acceder a estas propiedades desde el objeto Form, debe utilizarse la sintaxis *formInstance.propertyName*.

Propiedad	Descripción
`Loader.autoLoad`	Valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso llamar a `Loader.load()` (`false`).
`Loader.bytesLoaded`	Propiedad de sólo lectura que indica el número de bytes que se han cargado.
`Loader.bytesTotal`	Propiedad de sólo lectura que indica el número total de bytes del contenido.
`Loader.content`	Referencia al contenido del componente Loader. Es una propiedad de sólo lectura.
`Loader.contentPath`	Cadena que indica la URL del contenido que va a cargarse.
`Loader.percentLoaded`	Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.
`Loader.scaleContent`	Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader (`true`) o si el componente Loader se redimensiona para adaptarse al contenido (`false`).

Propiedades heredadas de la clase Screen

En la tabla siguiente se enumeran las propiedades que hereda la clase Form de la clase Screen. Al acceder a estas propiedades desde el objeto Form, debe utilizarse la sintaxis *formInstance.propertyName*.

Propiedad	Descripción
`Screen.currentFocusedScreen`	Sólo lectura; devuelve la pantalla que contiene la selección actual global.
`Screen.indexInParent`	Sólo lectura; devuelve el índice de la pantalla (basado en cero) en la lista de pantallas secundarias de su pantalla principal.
`Screen.numChildScreens`	Sólo lectura; devuelve el número de pantallas secundarias que contiene la pantalla.

Propiedad	Descripción
Screen.parentIsScreen	Sólo lectura; devuelve un valor booleano (true o false) que indica si el objeto principal de la pantalla también es una pantalla.
Screen.rootScreen	Sólo lectura; devuelve la pantalla raíz del árbol o subárbol que contiene la pantalla.

Resumen de eventos de la clase Form

No hay eventos exclusivos de la clase Form.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Form de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Form de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase Loader

En la tabla siguiente se enumeran los eventos que hereda la clase Form de la clase Loader.

Evento	Descripción
`Loader.complete`	Se activa cuando el contenido termina de cargarse.
`Loader.progress`	Se activa mientras se carga el contenido.

Eventos heredados de la clase Screen

En la tabla siguiente se enumeran los eventos que hereda la clase Form de la clase Screen.

Evento	Descripción
`Screen.allTransitionsInDone`	Se difunde cuando todas las transiciones de entrada aplicadas a una pantalla han finalizado.
`Screen.allTransitionsOutDone`	Se difunde cuando todas las transiciones de salida aplicadas a una pantalla han finalizado.
`Screen.mouseDown`	Se difunde cuando se ha presionado el botón del ratón sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
`Screen.mouseDownSomewhere`	Se difunde cuando se ha presionado el botón del ratón en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.
`Screen.mouseMove`	Se difunde cuando el ratón se mueve mientras se encuentra sobre una pantalla.
`Screen.mouseOut`	Se difunde cuando el ratón se mueve del interior al exterior de la pantalla.
`Screen.mouseOver`	Se difunde cuando el ratón se mueve del exterior al interior de la pantalla.
`Screen.mouseUp`	Se difunde cuando el botón del ratón se ha soltado sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
`Screen.mouseUpSomewhere`	Se difunde cuando el botón del ratón se ha soltado en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.

Form.currentFocusedForm

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
mx.screens.Form.currentFocusedForm
```

Descripción

Propiedad (sólo lectura); devuelve el objeto Form que contiene la selección actual global. La selección actual puede estar en el propio formulario o en un clip de película, objeto de texto o componente que se halle dentro de dicho formulario. Puede devolver el valor `null` en caso de que no haya selección actual.

Ejemplo

El código siguiente, asociado con un botón (que no se muestra), muestra el nombre del formulario que contiene la selección actual.

```
trace("The form with the current focus is: " +
  mx.screens.Form.currentFocusedForm);
```

Form.getChildForm()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
myForm.getChildForm(childIndex)
```

Parámetros

`childIndex` Número que indica el índice basado en cero del formulario secundario que debe devolverse.

Valor devuelto

Un objeto Form.

Descripción

Método; devuelve el formulario secundario de *myForm* cuyo índice es *childIndex*.

Ejemplo

En el ejemplo siguiente se muestra un panel Salida con los nombres de todos los objetos Form secundarios que pertenecen al objeto Form raíz denominado `application`.

```
for (var i:Number = 0; i < _root.application.numChildForms; i++) {
    var childForm:mx.screens.Form = _root.application.getChildForm(i);
    trace(childForm._name);
}
```

Véase también

`Form.numChildForms`

Form.indexInParentForm

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myForm`.indexInParentForm`

Descripción

Propiedad (sólo lectura); contiene el índice basado en cero de *myForm* en la lista de formularios secundarios de su formulario principal. Si el objeto principal de *myForm* es una pantalla en lugar de un formulario (por ejemplo, si es una diapositiva), `indexInParentForm` es siempre 0.

Ejemplo

```
var myIndex:Number = myForm.indexInParent;
if (myForm == myForm._parent.getChildForm(myIndex)) {
  trace("I'm where I should be");
}
```

Véase también

`Form.getChildForm()`

Form.numChildForms

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myForm.numChildForms

Descripción

Propiedad (sólo lectura); número de formularios secundarios que contiene *myForm* y que se derivan directamente de la clase mx.screens.Form. Esta propiedad no incluye diapositivas contenidas en *myForm*; sólo contiene formularios.

NOTA	Al usar una clase personalizada de ActionScript 2.0 que amplía mx.screens.Form, el formulario no se cuenta en la propiedad numChildForms.

Ejemplo

El código siguiente se repite en todos los formularios secundarios que contiene myForm y muestra sus nombres en el panel Salida.

```
var howManyKids:Number = myForm.numChildForms;
for(i=0; i<howManyKids; i++) {
  var childForm = myForm.getChildForm(i);
  trace(childForm._name);
}
```

Véase también

Form.getChildForm()

Form.parentIsForm

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myForm.parentIsForm

Descripción

Propiedad (sólo lectura): devuelve un valor booleano que indica si el objeto principal del formulario especificado también es un formulario (`true`) o si no lo es (`false`). Si esta propiedad es `false`, *myForm* se encuentra en la raíz de su jerarquía de formularios.

Ejemplo

```
if (myForm.parentIsForm) {
  trace("I have "+myForm._parent.numChildScreens+" sibling screens");
} else {
  trace("I am the root form and have no siblings");
}
```

Form.parentForm

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myForm.parentForm

Descripción

Propiedad (sólo lectura): referencia al formulario principal del formulario.

Ejemplo

El siguiente código de ejemplo reside en una pantalla denominada `myForm` que es una pantalla secundaria de la pantalla `form1` predeterminada, creada al elegir Aplicación de formularios Flash en el cuadro de diálogo Nuevo documento.

```
onClipEvent(keyDown){
  var parentForm:mx.screens.Form = this.parentForm;
  trace(parentForm);
}
// salida: _level0.application.form1
```

Form.rootForm

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`myForm``.rootForm`

Descripción

Propiedad (sólo lectura); devuelve el formulario superior de la jerarquía de formularios que contiene *`myForm`*. Si *`myForm`* se encuentra en un objeto que no es un formulario (es decir, una diapositiva), la propiedad devuelve *`myForm`*.

Ejemplo

En el ejemplo siguiente, se coloca una referencia al formulario raíz de `myForm` en una variable denominada `root`. Si el valor asignado a `root` hace referencia a `myForm`, `myForm` se encuentra en la parte superior de su árbol de formularios.

```
var root:mx.screens.Form = myForm.rootForm;
if(root == myForm) {
  trace("myForm is the top form in its tree");
}
```

Form.visible

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myForm.visible

Descripción

Propiedad; determina si *myForm* está visible cuando se puede ver su formulario, diapositiva, clip de película o archivo SWF principal. También puede establecer esta propiedad mediante el inspector de propiedades del entorno de edición de Flash.

Si esta propiedad tiene el valor `true`, *myForm* recibe un evento `reveal`; si tiene el valor `false`, *myForm* recibe un evento `hide`. Puede asociar transiciones con los formularios para que se ejecuten cuando un formulario reciba uno de estos eventos. Para más información sobre cómo añadir transiciones a las pantallas, consulte "Creación de controles y transiciones para las pantallas con comportamientos (sólo en Flash Professional)" en *Utilización de Flash*.

Ejemplo

El código siguiente, en un fotograma de la línea de tiempo, establece en `false` la propiedad `visible` del formulario que contiene el botón.

```
btnOk.addEventListener("click", btnOkClick);
function btnOkClick(eventObj:Object):Void {
  eventObj.target._parent.visible = false;
}
```

CAPÍTULO 25

Interfaz Iterator (sólo en Flash Professional)

25

Nombre de clase de ActionScript mx.utils.Iterator

La interfaz Iterator permite avanzar por los objetos que contiene una colección.

Resumen de métodos de la interfaz Iterator

En la tabla siguiente se enumeran los métodos de la interfaz Iterator.

Método	Descripción
`Iterator.hasNext()`	Indica si el repetidor tiene más elementos.
`Iterator.next()`	Devuelve el siguiente elemento de la repetición.

Iterator.hasNext()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

iterator`.hasNext()`

Valor devuelto

Valor booleano que indica si hay (`true`) o no hay (`false`) más instancias en el repetidor.

Descripción

Método; indica si hay más instancias en el repetidor. Este método se suele utilizar en una sentencia `while` al reproducir indefinidamente a través de un repetidor.

Ejemplo

El ejemplo siguiente utiliza el método `hasNext()` para controlar la reproducción indefinida a través del repetidor de elementos de una colección:

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDisks;
  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisk(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;
    trace("Title: "+title+" Artist: "+artist);
  }
}
```

Iterator.next()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

iterator.next()

Valor devuelto

Objeto que constituye el siguiente elemento del repetidor.

Descripción

Método; devuelve una instancia del siguiente elemento del repetidor. Esta instancia debe convertirse al tipo correcto.

Ejemplo

El ejemplo indicado a continuación utiliza el método `next()` para acceder al siguiente elemento de una colección:

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDisks;
  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisk(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;
    trace("Title: "+title+" Artist: "+artist);
  }
}
```

CAPÍTULO 26

Componente Label

26

Un componente Label es una línea de texto. Es posible especificar que una etiqueta se formatee con HTML. También se puede controlar la alineación y el tamaño de una etiqueta. Los componentes Label no tienen bordes, no se pueden seleccionar y no difunden eventos.

La previsualización dinámica de cada instancia de Label refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes. La etiqueta no tiene bordes, por lo que el único modo de obtener una previsualización dinámica es definir su parámetro text. El parámetro autoSize no se admite en la previsualización dinámica.

Cuando se añade el componente Label a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.LabelAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, “Creación de contenido accesible” en *Utilización de Flash*.

Utilización del componente Label

Utilice el componente Label para crear etiquetas de texto para otros componentes de un formulario, como la etiqueta “Nombre:” situada a la izquierda de un campo TextInput que acepta el nombre del usuario. Si está creando una aplicación con componentes basados en la versión 2 de la arquitectura de componentes de Macromedia, es recomendable utilizar un componente Label en lugar de un campo de texto normal, ya que así podrá utilizar estilos para lograr una apariencia uniforme.

Si desea rotar un componente Label, debe incorporar las fuentes. Véase “Utilización de estilos con el componente Label” en la página 782.

Parámetros de Label

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Label en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

autoSize indica cómo se cambia de tamaño y se alinea la etiqueta para que encaje el texto. El valor predeterminado es `none`. El parámetro puede tener cualquiera de estos cuatro valores:

- `none`, que especifica que la etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left`, que especifica que los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center`, que especifica que los lados izquierdo y derecho de la etiqueta cambian de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right`, que especifica que los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

NOTA

La propiedad `autoSize` del componente Label es diferente de la propiedad `autoSize` del objeto TextField incorporado de ActionScript.

html indica si la etiqueta va a formatearse con HTML (`true`) o no (`false`). Si este parámetro se establece como `true`, no se podrá formatear con estilos una etiqueta, pero sí se podrá aplicar formato al texto como HTML mediante la etiqueta `font`. El valor predeterminado es `false`.

text indica el texto de la etiqueta; el valor predeterminado es `Label`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente Label en el inspector de componentes (Ventana > Inspector de componentes):

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA

Las propiedades `minHeight` y `minWidth` se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Label con sus métodos, propiedades y eventos. Para más información, consulte “Clase Label” en la página 783.

Creación de aplicaciones con el componente Label

El siguiente procedimiento explica cómo añadir un componente Label a una aplicación durante la edición. En este ejemplo, la etiqueta se encuentra junto a un cuadro combinado de fechas en una aplicación de carrito de la compra.

Para crear una aplicación con el componente Label:

1. Arrastre un componente Label desde el panel Componentes al escenario.
2. En el inspector de componentes, introduzca **Expiration Date** en el parámetro de la etiqueta.

Para crear una instancia del componente Label mediante código ActionScript:

1. Arrastre el componente Label desde el panel Componentes a la biblioteca del documento actual.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.
2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

   ```
   this.createClassObject(mx.controls.Label, "my_label", 1);
   my_label.text = "Hello World";
   ```

   Este script utiliza el método `UIObject.createClassObject()` para crear la instancia de Label.
3. Seleccione Control > Probar película.

Personalización del componente Label

El componente Label puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. También es posible definir el parámetro de edición autoSize; en tal caso, el recuadro de delimitación de la previsualización dinámica no varía, pero la etiqueta sí cambia de tamaño. Para más información, consulte "Parámetros de Label" en la página 780. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()` o `Label.autoSize`).

Utilización de estilos con el componente Label

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de Label. Todo el texto de la instancia del componente Label debe compartir el mismo estilo. Por ejemplo, no es posible definir el estilo `color` en `"blue"` para una palabra de la etiqueta y en `"red"` para la segunda palabra de la misma etiqueta.

Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente Label admite los siguientes estilos:

Estilo	Tema	Descripción
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

Utilización de aspectos con el componente Label

El componente Label no tiene elementos visuales a los que aplicar aspectos.

Clase Label

Herencia MovieClip > Clase UIObject > Label

Nombre de clase de ActionScript mx.controls.Label

Las propiedades de la clase Label permiten especificar en tiempo de ejecución el texto de la etiqueta, indicar si el texto se va a formatear con HTML e indicar si la etiqueta cambia de tamaño automáticamente para que encaje el texto.

Si una propiedad de la clase Label se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Cuando acceda a los valores de las propiedades de etiqueta, asegúrese de que el componente se ha cargado completamente antes de intentar acceder a la propiedad que desee. Observe el siguiente ejemplo:

```
var listenerObject:Object = new Object();
listenerObject.load = function(){
  trace(label.width);
};
label.addEventListener("load", listenerObject);
```

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.Label.version);
```

NOTA El código `trace(myLabelInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase Label

No hay métodos exclusivos de la clase Label.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Label de la clase UIObject. Al llamar a estos métodos desde el objeto Label, debe utilizarse la forma *`labelInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Resumen de propiedades de la clase Label

En la tabla siguiente se enumeran las propiedades de la clase Label.

Propiedad	Descripción
`Label.autoSize`	Cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad `text`. Hay cuatro valores posibles: `"none"`, `"left"`, `"center"` y `"right"`. El valor predeterminado es `"none"`.
`Label.html`	Valor booleano que indica si la etiqueta puede formatearse con HTML (`true`) o no (`false`).
`Label.text`	Texto de la etiqueta.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Label de la clase UIObject. Al acceder a estas propiedades, debe utilizarse la forma *`labelInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.

Propiedad	Descripción
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Resumen de eventos de la clase Label

No hay eventos exclusivos de la clase Label.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Label de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Label.autoSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

labelInstance.autoSize

Descripción

Propiedad; cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad `text`. Hay cuatro valores posibles: `"none"`, `"left"`, `"center"` y `"right"`. El valor predeterminado es `"none"`.

- `none` La etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left` Los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center` Los lados izquierdo y derecho de la etiqueta cambian de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right` Los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

> **NOTA** La propiedad `autoSize` del componente Label es diferente de la propiedad `autoSize` del objeto TextField incorporado de ActionScript.

Ejemplo

En el siguiente ejemplo, la instancia de Label `my_label` cambia el tamaño de los lados izquierdo e inferior de la etiqueta para que encaje el texto:

```
my_label.text = "A really long label with Label.autoSize set";
my_label.autoSize = "right";
```

Label.html

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`labelInstance``.html`

Descripción

Propiedad; valor booleano que indica si la etiqueta puede formatearse con HTML (`true`) o no (`false`). El valor predeterminado es `false`. Los componentes Label con la propiedad `html` definida en `true` no pueden formatearse con estilos.

Para recuperar el texto normal de un texto con formato HTML, defina la propiedad `HTML` en `false` y, a continuación, acceda a la propiedad `text`. Esto eliminará el formato HTML. Tal vez desee copiar el texto de la etiqueta en un componente Label o TextArea de fuera de la pantalla antes de recuperar el texto normal.

Ejemplo

En el ejemplo siguiente se define la propiedad `html` en `true` para que la etiqueta pueda formatearse con HTML. La propiedad `text` se define en una cadena que incluye formato con HTML.

```
my_label.html = true;
my_label.text = "The <b>Royal</b> Nonesuch";
my_label.autoSize = "right";
```

La palabra “Royal” se muestra en negrita.

Label.text

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`labelInstance``.text`

Descripción

Propiedad; texto de una etiqueta. El valor predeterminado es `"Label"`.

Ejemplo

El código siguiente define la propiedad `text` de la instancia `my_label` de Label y envía el valor al panel Salida:

```
my_label.text = "The Royal Nonesuch";
trace(my_label.text);
```

CAPÍTULO 27

Componente List

27

El componente List es un cuadro de lista de desplazamiento de selección única o múltiple. Una lista también puede mostrar gráficos, incluidos otros componentes. Los elementos que se visualizan en la lista se añaden con el cuadro de diálogo Valores que aparece al hacer clic en los campos de parámetros data o labels. Para añadir elementos a la lista también pueden utilizarse los métodos `List.addItem()` y `List.addItemAt()`.

El componente List utiliza un índice basado en cero, en el que el elemento con el índice 0 aparece en primer lugar. Si añade, elimina o reemplaza elementos de lista mediante los métodos y propiedades de la clase List, es posible que tenga que especificar el índice del elemento de lista.

La lista se selecciona cuando se hace clic en ella o se presiona el tabulador hasta su posición y, a continuación, se pueden utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Teclas alfanuméricas	Salta al siguiente elemento que tiene `Key.getAscii()` como el primer carácter de su etiqueta.
Control	Tecla conmutadora que permite seleccionar y anular la selección de varios elementos no contiguos.
Flecha abajo	La selección desciende un elemento.
Inicio	La selección se desplaza al principio de la lista.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Mayús	Permite seleccionar elementos contiguos.
Flecha arriba	La selección asciende un elemento.

NOTA

Para las teclas Av Pág y Re Pág, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte “Clase FocusManager” en la página 745 o “Creación de un desplazamiento personalizado de la selección” en *Utilización de componentes*.

La previsualización dinámica de cada instancia de List del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el inspector de componentes.

Cuando se añade el componente List a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ListAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, “Creación de contenido accesible” en *Utilización de Flash*.

Utilización del componente List

Se puede configurar una lista para que los usuarios efectúen selecciones únicas o múltiples. Por ejemplo, un usuario visita un sitio Web de comercio electrónico y debe elegir el artículo que va a comprar. Hay 30 artículos; el usuario recorre la lista y hace clic en el que desea seleccionar.

También se puede diseñar una lista que utilice clips de película personalizados como filas para poder mostrar más información al usuario. Por ejemplo, en una aplicación de correo electrónico, cada buzón podría ser un componente List y cada fila podría tener iconos para indicar la prioridad y el estado.

Introducción al diseño del componente List

Al diseñar una aplicación con el componente List o cualquier componente que amplíe la clase List, resulta útil entender cómo se ha diseñado la lista. A continuación, se detallan algunos requisitos y suposiciones fundamentales que Macromedia ha usado para desarrollar la clase List:

- Cuanto más pequeño, rápido y simple, mejor.

  Es preferible no hacer algo más complicado de lo que sea absolutamente necesario. Esta ha sido la directiva de diseño más importante; la mayoría de los requisitos que se enumeran a continuación se basan en esta directiva.

- Las listas tienen alturas de fila uniformes.

  Cada fila debe tener la misma altura; ésta se puede definir durante la edición o en tiempo de ejecución.

- La escala de las listas debe ajustarse a miles de registros.
- Las listas no miden el texto.

  Esta restricción tiene las ramificaciones con mayor potencial. Puesto que la escala de una lista debe ajustarse a miles de registros, de los cuales cualquiera podría contener una cadena excepcionalmente larga, dicha lista no debe crecer para ajustar la mayor cadena de texto dentro de ella ni añadir una barra de desplazamiento horizontal en el modo "auto". Asimismo, medir miles de cadenas resultaría demasiado intensivo. La solución es la propiedad `maxHPosition`, que, cuando se establece `vScrollPolicy` en `"on"`, proporciona el espacio de búfer adicional de la lista para el desplazamiento.

  Si prevé la aparición de cadenas largas, establezca `hScrollPolicy` en `"on"` y añada un valor de 200 píxeles `maxHPosition` al componente List o Tree. Más o menos se garantiza que el usuario podrá desplazarse para verlo todo. Sin embargo, el componente DataGrid admite `"auto"` como valor `hScrollPolicy`, ya que mide columnas (con la misma anchura por elemento) y no texto.

  El hecho de que las listas no midan texto también explica por qué las listas tienen alturas de fila uniformes. Cambiar el tamaño de cada fila para que se ajuste al texto precisa demasiadas mediciones. Por ejemplo, si desea mostrar de forma precisa las barras de desplazamiento de una lista con altura de fila no uniforme, no es necesario medir previamente cada fila.
- Las listas funcionan peor como función de sus filas visibles.

  Aunque hay listas que pueden mostrar 5000 registros, no pueden representar 5000 registros a la vez. Cuantas más filas visibles (especificadas por la propiedad `rowCount`) tenga en el escenario, más trabajo deberá llevar a cabo la lista para generar sus representaciones. Limitar el número de filas visibles, si es posible, supone la mejor solución.
- Las listas no son tablas.

  Por ejemplo, los componentes DataGrid, que amplían la clase List, están pensados para proporcionar una interfaz para muchos registros. No están diseñados para mostrar toda la información, sino para mostrar la suficiente información como para que los usuarios puedan ampliarla posteriormente. La vista de mensajes en Microsoft Outlook es un excelente ejemplo. No se lee todo el mensaje de correo electrónico en la cuadrícula; resultaría difícil leerlo y el rendimiento del cliente sería bajo. Outlook muestra suficiente información para que el usuario vaya al elemento expuesto y vea más detalles.

Parámetros de List

A continuación, se indican los parámetros de edición que se pueden definir para cada instancia del componente List en el inspector de propiedades o en el inspector de componentes:

data es una matriz de valores que rellenan los datos de la lista. El valor predeterminado es `[]` (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

labels es una matriz de valores de texto que rellenan los valores de etiqueta de la lista. El valor predeterminado es `[]` (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

multipleSelection es una valor booleano que indica si se permite la selección de varios valores (`true`) o no (`false`). El valor predeterminado es `false`.

rowHeight indica la altura de cada fila, expresada en píxeles. El valor predeterminado es 20. Si se define una fuente, la altura de la fila no cambia.

Puede escribir código ActionScript para definir opciones adicionales para instancias de List con sus métodos, propiedades y eventos. Para más información, consulte "Clase List" en la página 799.

Creación de aplicaciones con el componente List

El siguiente procedimiento explica cómo añadir un componente List a una aplicación durante la edición. En este ejemplo, la lista tiene tres elementos.

Para añadir un componente List a una aplicación:

1. Arrastre un componente List desde el panel Componentes al escenario.
2. Seleccione la herramienta Transformación libre y cambie el tamaño del componente para que se ajuste a la aplicación.
3. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de la instancia **my_list**.
   - Introduzca **Item1**, **Item2** e **Item3** como parámetro labels.
   - Introduzca **item1.html**, **item2.html** e **item3.html** como parámetro data.
4. Elija Control > Probar película para ver la lista con sus elementos.
5. Vuelva al entorno de edición, inserte una nueva capa y asígnele el nombre **acciones**.
6. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones.

```
my_list.change = function(evt:Object) {
  getURL(evt.target.selectedItem.data, "_blank");
};
my_list.addEventListener("change", my_list);
```

Para rellenar una instancia de List con un proveedor de datos:

1. Arrastre un componente List desde el panel Componentes al escenario.
2. Seleccione la herramienta Transformación libre y cambie el tamaño del componente para que se ajuste a la aplicación.
3. En el inspector de propiedades, introduzca el nombre de instancia **my_list**.
4. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca lo siguiente:

   ```
   my_list.dataProvider = myDP;
   ```

   Si ha definido un proveedor de datos denominado `myDP`, la lista se llenará de datos. Para más información sobre los proveedores de datos, consulte `List.dataProvider`.
5. Elija Control > Probar película para ver la lista con sus elementos.

Para utilizar un componente List que permita controlar un clip de película

1. Arrastre un componente List desde el panel Componentes al escenario.
2. Seleccione la herramienta Transformación libre y cambie el tamaño del componente para que se ajuste a la aplicación.
3. En el inspector de propiedades, introduzca el nombre de instancia **my_list**.
4. Cree un clip de película en el escenario y asígnele el nombre de instancia **my_mc**.
5. Abra el clip de película en el modo de edición de símbolos y añada alguna animación.
6. Inserte una nueva capa y asígnele el nombre **acciones**.
7. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones.

   ```
   my_list.addItem({label:"play", data:"play"});
   my_list.addItem({label:"stop", data:"stop"});
   var listHandler:Object = new Object();
   listHandler.change = function(evt:Object) {
     switch (evt.target.selectedItem.data) {
     case "play" :
       my_mc.play();
       break;
     case "stop" :
       my_mc.stop();
       break;
     default :
       trace("unhandled event: "+evt.target.selectedItem.data);
       break;
     }
   };
   my_list.addEventListener("change", listHandler);
   ```
8. Seleccione Control > Probar película para que la lista reproduzca y detenga la instancia del clip de película `my_mc`.

Para crear una instancia del componente List mediante código ActionScript:

1. Arrastre un componente List desde el panel Componentes a la biblioteca.

   De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

   ```
   this.createClassObject(mx.controls.List, "my_list", 1);
   my_list.addItem({label:"One", data:dt1});
   my_list.addItem({label:"Two", data:dt2});
   ```

   Este script utiliza el método `UIObject.createClassObject()` para crear la instancia de List.

3. Seleccione Control > Probar película.

Personalización del componente List

El componente List puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `List.setSize()` (véase `UIObject.setSize()`).

Cuando se cambia el tamaño de una lista, las filas se contraen horizontalmente y recortan el texto que contienen. Verticalmente, la lista añade o elimina las filas necesarias. Las barras de desplazamiento se colocan automáticamente. Para más información sobre barras de desplazamiento, consulte “Componente ScrollPane” en la página 1131.

Utilización de estilos con el componente List

Es posible definir propiedades de estilo para cambiar el aspecto de un componente List.

El componente List utiliza los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`alternatingRowColors`	Ambos	Especifica los colores de las filas en un patrón alterno. El valor puede ser una matriz de más de dos colores, por ejemplo, 0xFF00FF, 0xCC6699 y 0x996699. A diferencia de los estilos de color de un solo valor, `alternatingRowColors` no acepta nombres de color; los valores deben ser códigos de color numéricos. De forma predeterminada, el estilo no está definido y `backgroundColor` se usa en su lugar para todas las filas.
`backgroundColor`	Ambos	Color de fondo de la lista. El color predeterminado es blanco y se define en la declaración de estilo de clase. Este estilo se omite si se especifica `alternatingRowColors`.
`backgroundDisabledColor`	Ambos	Color de fondo cuando la propiedad del componente `enabled` está establecida en `"false"`. El valor predeterminado es 0xDDDDDD (gris medio).
`borderStyle`	Ambos	El componente List utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es `"inset"`.
`color`	Ambos	Color del texto.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.

Estilo	Tema	Descripción
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.
`defaultIcon`	Ambos	Nombre del icono predeterminado que se mostrará en cada fila. El valor predeterminado es `undefined`, el cual indica que no se muestra ningún icono.
`repeatDelay`	Ambos	Número de milisegundos de demora entre el momento en que el usuario presiona por primera vez un botón en la barra de desplazamiento y el momento en que comienza a repetirse la acción. El valor predeterminado es 500, medio segundo.
`repeatInterval`	Ambos	Número de milisegundos entre clics automáticos cuando un usuario mantiene presionado el botón del ratón sobre un botón de la barra de desplazamiento. El valor predeterminado es 35.
`rollOverColor`	Ambos	Color de fondo de una fila por la que se desplaza el puntero. El valor predeterminado es 0xE3FFD6 (verde brillante) con el tema Halo y 0xAAAAAA (gris claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a `setStyle()`, el marco establece `rollOverColor` en un valor relacionado con el valor de `themeColor` elegido.

Estilo	Tema	Descripción
`selectionColor`	Ambos	Color de fondo de una fila seleccionada. El valor predeterminado es 0xCDFFC1 (verde claro) con el tema Halo y 0xEEEEEE (gris muy claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a `setStyle()`, el marco establece `selectionColor` en un valor relacionado con el valor de `themeColor` elegido.
`selectionDuration`	Ambos	Duración de la transición del estado normal al estado seleccionado o viceversa, expresado en milisegundos. El valor predeterminado es 200.
`selectionDisabledColor`	Ambos	Color de fondo de una fila seleccionada. El valor predeterminado es 0xDDDDDD (gris medio). Como el valor predeterminado de esta propiedad coincide con el valor predeterminado de `backgroundDisabledColor`, la selección no es visible cuando se desactiva el componente, a menos que se cambie una de estas propiedades de estilo.
`selectionEasing`	Ambos	Referencia a la ecuación de suavizado que se utiliza para controlar la transición entre los estados de selección. Esto sólo se aplica a la transición del estado normal al seleccionado. La ecuación predeterminada usa una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.
`textRollOverColor`	Ambos	Color del texto cuando el puntero se desplaza sobre él. El valor predeterminado es 0x2B333C (gris oscuro). Este estilo es importante cuando se establece `rollOverColor` porque ambos colores deben complementarse para que el texto pueda verse con claridad al desplazar el puntero sobre él.
`textSelectedColor`	Ambos	Color del texto de la fila seleccionada. El valor predeterminado es 0x005F33 (gris oscuro). Este estilo es importante al definir `selectionColor`, ya que los dos valores deben complementarse entre sí para que se pueda ver fácilmente el texto cuando esté seleccionado.
`useRollOver`	Ambos	Determina si el resaltado se activa cuando el puntero se desplaza sobre una fila. El valor predeterminado es `true`.

Definición de estilos para todos los componentes List de un documento

La clase List hereda de la clase ScrollSelectList. Las propiedades de estilo de clase predeterminadas se definen en la clase ScrollSelectList, que amplían el componente Menu y todos los componentes basados en List. Puede definir directamente en esta clase nuevos valores de estilo predeterminados y los nuevos valores se reflejan en todos los componentes afectados.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Para definir una propiedad de estilo únicamente en los componentes List y los componentes basados en List, se puede crear una nueva instancia de `CSSStyleDeclaration` y almacenarla en `_global.styles.List`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.List == undefined) {
  _global.styles.List = new CSSStyleDeclaration();
}
_global.styles.List.setStyle("backgroundColor", 0xFF00AA);
```

Cuando se crea una nueva declaración de estilo de clase, se pierden todos los valores predeterminados proporcionados por la declaración `ScrollSelectList`. Entre ellos, el valor de `backgroundColor`, necesario para la compatibilidad con eventos de ratón. Para crear una declaración de estilo de clase y conservar los valores predeterminados, utilice un bucle `for..in` para copiar los valores anteriores a la nueva declaración.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.List;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Para proporcionar estilos al componente List, pero no a componentes que amplíen List (DataGrid y Tree), debe ofrecer declaraciones de estilo de clase a estas subclases.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.DataGrid == undefined) {
  _global.styles.DataGrid = new CSSStyleDeclaration();
}
_global.styles.DataGrid.setStyle("backgroundColor", 0xFFFFFF);
if (_global.styles.Tree == undefined) {
  _global.styles.Tree = new CSSStyleDeclaration();
}
_global.styles.Tree.setStyle("backgroundColor", 0xFFFFFF);
```

Para más información sobre los estilos de clase, consulte “Definición de los estilos de una clase de componente” en *Utilización de componentes*.

Utilización de aspectos con el componente List

El componente List utiliza una instancia de RectBorder para su borde y barras de desplazamiento para desplazarse por las imágenes. Para más información sobre la aplicación de aspectos en estos elementos visuales, consulte "Clase RectBorder" en la página 1103 y "Utilización de aspectos con el componente UIScrollBar" en la página 1435.

Clase List

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > ScrollSelectList > List

Nombre de clase de ActionScript mx.controls.List

El componente List consta de tres partes: elementos, fila y proveedor de datos.

Un *elemento* es un objeto de ActionScript que sirve para almacenar las unidades de información de la lista. Una lista puede concebirse como una matriz; cada espacio indexado de la matriz es un elemento. Un elemento es un objeto que, normalmente, tiene una propiedad `label` que se muestra y una propiedad `data` que se utiliza para almacenar datos.

Una *fila* es un componente que se utiliza para mostrar un elemento. Las filas las suministra de forma predeterminada la lista (se utiliza la clase SelectableRow) o las suministra el usuario, normalmente con una subclase de la clase SelectableRow. La clase SelectableRow implementa la interfaz API CellRenderer, que es el conjunto de propiedades y métodos que permiten que la lista pueda manipular cada fila y enviar datos e información sobre el estado (por ejemplo, tamaño, selección) a la fila para su visualización.

Un proveedor de datos es el modelo de datos de la lista de elementos de una lista. Si en un mismo fotograma hay una lista y una matriz, ésta recibe automáticamente métodos que permiten manipular datos y difundir cambios a varias vistas. Puede crear una instancia de Array u obtener una en un servidor y utilizarla como modelo de datos para varias listas, cuadros combinados, cuadrículas de datos, etc. El componente List posee métodos que le permiten comunicarse con su proveedor de datos (por ejemplo, `addItem()` y `removeItem()`). Si no se proporciona ningún proveedor de datos externo a la lista, estos métodos crean automáticamente una instancia de proveedor de datos que se presenta mediante `List.dataProvider`.

Para añadir un componente List al orden de tabulación de una aplicación, defina su propiedad `tabIndex` (véase `UIComponent.tabIndex`). El componente List utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.List.version);
```

NOTA

El código `trace(myListInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase List

En la tabla siguiente se enumeran los métodos de la clase List.

Método	Descripción
`List.addItem()`	Añade un elemento al final de la lista.
`List.addItemAt()`	Añade un elemento a la lista en el índice especificado.
`List.getItemAt()`	Devuelve el elemento del índice especificado.
`List.removeAll()`	Elimina todos los elementos de la lista.
`List.removeItemAt()`	Elimina el elemento en el índice especificado.
`List.replaceItemAt()`	Sustituye por otro el elemento del índice especificado.
`List.setPropertiesAt()`	Aplica las propiedades especificadas al elemento especificado.
`List.sortItems()`	Ordena los elementos de la lista según la función de comparación especificada.
`List.sortItemsBy()`	Ordena los elementos de la lista según la propiedad especificada.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase List de la clase UIObject. Al llamar a estos métodos, debe utilizarse la forma *`listInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.

Método	Descripción
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase List de la clase UIComponent. Al llamar a estos métodos, debe utilizarse la forma *listInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase List

En la tabla siguiente se enumeran las propiedades de la clase List.

Propiedad	Descripción
List.cellRenderer	Asigna la clase o el símbolo que se va utilizar para mostrar cada fila de la lista.
List.dataProvider	Origen de los elementos de la lista.
List.hPosition	Posición horizontal de la lista.
List.hScrollPolicy	Indica si la barra de desplazamiento horizontal se muestra ("on") o no ("off").

Propiedad	Descripción
`List.iconField`	Campo de cada elemento que se utiliza para especificar iconos.
`List.iconFunction`	Función que determina qué icono se debe utilizar.
`List.labelField`	Especifica el campo de cada elemento que se utilizará como texto de etiqueta.
`List.labelFunction`	Función que determina qué campos de cada elemento se utilizarán para el texto de etiqueta.
`List.length`	Número de elementos de la lista. Es una propiedad de sólo lectura.
`List.maxHPosition`	Número de píxeles que la lista puede desplazarse a la derecha cuando `List.hScrollPolicy` está definida en `"on"`.
`List.multipleSelection`	Indica si en la lista se permite la selección múltiple (`true`) o no (`false`).
`List.rowCount`	Número de filas que son al menos parcialmente visibles en la lista.
`List.rowHeight`	Altura de las filas de la lista, expresada en píxeles.
`List.selectable`	Indica si la lista es seleccionable (`true`) o no (`false`).
`List.selectedIndex`	Índice de una selección en una lista de selección única.
`List.selectedIndices`	Matriz de los elementos seleccionados en una lista de selección múltiple.
`List.selectedItem`	Elemento seleccionado en una lista de selección única. Es una propiedad de sólo lectura.
`List.selectedItems`	Objetos del elemento seleccionado en una lista de selección múltiple. Es una propiedad de sólo lectura.
`List.vPosition`	Elemento visible situado en la parte superior de la lista.
`List.vScrollPolicy`	Indica si la barra de desplazamiento vertical se muestra (`"on"`), no se muestra (`"off"`) o se muestra cuando es necesario (`"auto"`).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase List de la clase UIObject. Al acceder a estas propiedades, debe utilizarse la forma *listInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
UIObject.height	Altura del objeto, expresada en píxeles. Sólo lectura.
UIObject.left	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase List de la clase UIComponent. Al acceder a estas propiedades, debe utilizarse la forma *listInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase List

En la tabla siguiente se enumeran los eventos de la clase List.

Evento	Descripción
`List.change`	Se difunde cuando la interacción del usuario provoca un cambio en la selección.
`List.itemRollOut`	Se difunde cuando el puntero se desplaza sobre elementos de la lista y después sale de ellos.
`List.itemRollOver`	Se difunde cuando el puntero se desplaza sobre un elemento.
`List.scroll`	Se difunde cuando se recorre la lista.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase List de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase List de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

List.addItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listInstance.addItem(label[, data])
listInstance.addItem(itemObject)
```

Parámetros

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es optativo y puede ser de cualquier tipo de datos.

itemObject Objeto de elemento que normalmente tiene propiedades `label` y `data`.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad `label` especificada y, si se ha especificado, la propiedad `data`.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

Las dos siguientes líneas de código añaden un elemento a la instancia `my_list`. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.addItem("this is an Item");
my_list.addItem({label:"Gordon", age:"very old", data:123});
```

List.addItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.addItemAt(*index*, *label*[, *data*])

listInstance.addItemAt(*index*, *itemObject*)

Parámetros

index Número mayor o igual que 0 que indica la posición del elemento.

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es optativo y puede ser de cualquier tipo de datos.

itemObject Objeto de elemento que normalmente tiene propiedades `label` y `data`.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo en la posición especificada por el parámetro *index*.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad `label` especificada y, si se ha especificado, la propiedad `data`.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

En el siguiente ejemplo se añade un elemento en la posición del primer índice, que es el segundo elemento de la lista. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.addItem("this is an Item");
my_list.addItem({label:"Gordon", age:"very old", data:123});
my_list.addItemAt(1, {label:"Red", data:0xFF0000});
```

List.cellRenderer

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listInstance.cellRenderer
```

Descripción

Propiedad; asigna el procesador de celdas que va a utilizarse en cada fila de la lista. Esta propiedad debe ser una referencia a un objeto de clase o un identificador de vinculación del símbolo. Todas las clases que se utilicen con esta propiedad deben implementar Interfaz API CellRenderer.

Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de celdas:

```
my_list.cellRenderer = "ComboBoxCell";
```

List.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // El código se escribe aquí.
};
listInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // El código se escribe aquí.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el índice seleccionado de la lista cambia como resultado de la interacción del usuario.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, change), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher (API)" en la página 516.

Finalmente, se llama al método addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

El segundo ejemplo de sintaxis utiliza un controlador on() que debe asociarse directamente con una instancia de List. La palabra clave this, utilizada en un controlador on() asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia my_list del componente List, envía "_level0.my_list" al panel Salida:

```
on (change) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se añaden tres elementos al componente List. Si se cambia el valor seleccionado de la lista, el valor del elemento que ha seleccionado aparecerá en el panel Salida. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});

// Crear un objeto detector.
var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
   trace("Value changed to: " + evt_obj.target.value);
}
// Añadir detector.
my_list.addEventListener("change", listListener);
```

Véase también

`EventDispatcher.addEventListener()`

List.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`listInstance``.dataProvider`

Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la API DataProvider. El valor predeterminado es `[]`. Para más información, consulte "Interfaz API de DataProvider" en la página 333.

El componente List, al igual que otros componentes para datos, añade métodos al prototipo del objeto Array para ceñirse a la interfaz API DataProvider. Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de datos de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, accede a las propiedades `List.labelField` o `List.labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es `"label"`, por lo que si existe un campo `label`, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

NOTA Si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar, se pierde la selección.

Como proveedor de datos del componente List pueden elegirse todas las instancias que implementen la interfaz API DataProvider. Entre ellas se encuentran los juegos de registros de Flash Remoting, los juegos de datos de Firefly, etc.

Ejemplo

En el siguiente ejemplo se utiliza una matriz de cadenas para rellenar la lista:

```
my_list.dataProvider = ["Ground Shipping", "2nd Day Air", "Next Day Air"];
```

Este ejemplo crea una matriz de proveedor de datos y la asigna a la propiedad `dataProvider`, como se muestra a continuación:

```
var myDP_array:Array = new Array();
my_list.dataProvider = myDP_array;

var accounts_array:Array = new Array();
accounts_array.push({name:"checkings", accountID:12345});
accounts_array.push({name:"savings", accountID:67890});

for (var i:Number = 0; i < accounts_array.length; i++) {
 // Estos cambios en el proveedor de datos se difundirán a la lista
 myDP_array.addItem({label:accounts_array[i].name,
  data:accounts_array[i].accountID});
}
```

List.getItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`listInstance``.getItemAt(`*`index`*`)`

Parámetros

`index` Número mayor o igual que 0 y menor que `List.length`. Especifica el índice del elemento que debe recuperarse.

Valor devuelto

Objeto del elemento indexado; `undefined` si el índice está fuera de rango.

Descripción

Método; recupera el elemento del índice especificado. Este método obtiene el elemento de datos de una matriz, DataProvider, o de un objeto de datos creado mediante `CellRenderer.setValue()`.

Ejemplo

El siguiente código muestra la etiqueta del elemento en la posición de índice 2. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});

trace(my_list.getItemAt(2).label);
```

List.hPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.hPosition

Descripción

Propiedad; desplaza la lista el número de píxeles especificado en sentido horizontal. `hPosition` no puede definirse a menos que el valor de `hScrollPolicy` sea `"on"` y la lista tenga un `maxHPosition` superior a 0.

Ejemplo

El siguiente código muestra el valor actual de `hPosition` siempre que la instancia de List se recorra de forma horizontal. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 50;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});
```

```
var listListener:Object = new Object();
listListener.scroll = function (evt_obj:Object) {
 trace("my_list.hPosition = " + my_list.hPosition);
}
my_list.addEventListener("scroll", listListener);
```

List.hScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.hScrollPolicy

Descripción

Propiedad; cadena que determina si se muestra la barra de desplazamiento horizontal; el valor puede ser `"on"` u `"off"`. El valor predeterminado es `"off"`. La barra de desplazamiento horizontal no mide el texto; es preciso definir una posición máxima de desplazamiento horizontal (véase `List.maxHPosition`).

NOTA	`List.hScrollPolicy` no admite el valor `"auto"`.

Ejemplo

El código siguiente activa el desplazamiento horizontal de la lista hasta 200 píxeles. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 200;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});
```

Véase también

`List.hPosition`, `List.maxHPosition`

List.iconField

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.iconField

Descripción

Propiedad; especifica el nombre del campo que va a utilizarse como identificador de icono. Si el valor del campo es `undefined`, se utiliza el icono predeterminado especificado en el estilo `defaultIcon`. Si el estilo `defaultIcon` es `undefined`, no se utiliza ningún icono.

Ejemplo

En el ejemplo siguiente se define la propiedad `iconField` en la propiedad `icon` de cada elemento. Para probar este código, arrastre un componente List al escenario, asígnele el nombre de instancia **my_list** y cree tres símbolos con los nombres de instancia `flash`, `dreamweaver` y `cf` respectivamente. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente List en el escenario (nombre de instancia: my_list)
  - Símbolo MovieClip/Graphic en la biblioteca con identificación de
  vinculación de "flash"
  - Símbolo MovieClip/Graphic en la biblioteca con identificación de
  vinculación de "dreamweaver"
  - Símbolo MovieClip/Graphic en la biblioteca con identificación de
  vinculación de "cf"
*/

var my_list:mx.controls.List;

my_list.setSize(200, 100);

my_list.addItem({data:"flash", label:"Flash", icon:"flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver",
  icon:"dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion", icon:"cf"});

my_list.iconField = "icon";
```

Véase también

`List.iconFunction`

List.iconFunction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.iconFunction

Descripción

Propiedad; especifica la función que determina qué icono utilizará cada fila para mostrar su elemento. Esta función recibe un parámetro *item*, que es el elemento que se va a representar, y debe devolver una cadena que represente el identificador de símbolo del icono.

Ejemplo

En el ejemplo siguiente se añaden iconos que indican si el archivo es un documento de imagen o de texto. Si el campo `data.fileExtension` contiene un valor `"jpg"` o `"gif"`, el icono utilizado será `"pictureIcon"`, y así sucesivamente.

```
my_list.iconFunction = function(item:Object):String {
  var type:String = item.data.fileExtension;
  if (type == "jpg" || type == "gif") {
    return "pictureIcon";
  } else if (type == "doc" || type == "txt") {
    return "docIcon";
  }
}
```

En el ejemplo siguiente se define la propiedad `iconField` en la propiedad `icon` de cada elemento. Para probar este código, arrastre un componente List al escenario, asígnele el nombre de instancia **my_list** y cree tres símbolos con los nombres de instancia `flash`, `dreamweaver` y `cf` respectivamente. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente List en el escenario (nombre de instancia: my_list)
  - Símbolo MovieClip/Graphic en la biblioteca con identificación de
  vinculación de "flashIcon"
*/
```

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.iconFunction = function(item:Object):String {
  if (item.data == "flash") {
    // Colocar icono junto a elemento de lista con los datos "flash".
    return "flashIcon";
  }
};
my_list.iconField = "icon";
```

List.itemRollOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.itemRollOut = function(eventObject:Object) {
  // El código se escribe aquí.
};
listInstance.addEventListener("itemRollOut", listenerObject);
```

Sintaxis 2:

```
on (itemRollOut) {
  // El código se escribe aquí.
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `itemRollOut` tiene una propiedad `index`, que especifica el número del elemento del que ha salido el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza sobre elementos de la lista y después sale de ellos.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, `itemRollOut`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher (API)" en la página 516.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de List. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `my_list` del componente List, envía "_level0.my_list" al panel Salida:

```
on (itemRollOut) {
  trace(this);
}
```

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Crear un objeto detector.
var listListener:Object = new Object();
listListener.itemRollOut = function(evt_obj:Object) {
 trace("Item #" + evt_obj.index + " has been rolled out.");
};

// Añadir detector.
my_list.addEventListener("itemRollOut", listListener);
```

Véase también

`List.itemRollOver`

List.itemRollOver

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.itemRollOver = function(eventObject:Object) {
  // El código se escribe aquí.
};
listInstance.addEventListener("itemRollOver", listenerObject);
```

Sintaxis 2:

```
on (itemRollOver) {
  // El código se escribe aquí.
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `itemRollOver` tiene una propiedad `index`, que especifica el número del elemento sobre el que ha pasado el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza por los elementos de la lista.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, itemRollOver), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método EventDispatcher.addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher (API)" en la página 516.

El segundo ejemplo de sintaxis utiliza un controlador on() que debe asociarse directamente con una instancia de List. La palabra clave this, utilizada en un controlador on() asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia my_list del componente List, envía "_level0.my_list" al panel Salida:

```
on (itemRollOver) {
  trace(this);
}
```

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Crear un objeto detector.
var listListener:Object = new Object();
listListener.itemRollOver = function (evt_obj:Object) {
 trace("Item #" + evt_obj.index + " has been rolled over.");
};

// Añadir detector.
my_list.addEventListener("itemRollOver", listListener);
```

Véase también

List.itemRollOut

List.labelField

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.labelField

Descripción

Propiedad; especifica un campo en cada elemento que se utilizará como el texto que se muestra. Esta propiedad toma el valor del campo y lo utiliza como etiqueta. El valor predeterminado es `"label"`.

Ejemplo

En el ejemplo siguiente se define la propiedad `labelField` en el campo `"name"` de cada elemento. “Nina” aparecería como la etiqueta del elemento añadido en la segunda línea de código:

```
var my_list:mx.controls.List;

my_list.labelField = "name";
my_list.addItem({name: "Nina", age: 25});
```

Véase también

`List.labelFunction`

List.labelFunction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.labelFunction

Descripción

Propiedad; especifica una función que determina qué campo (o combinación de campos) de cada elemento va a mostrarse. Esta función recibe un parámetro *item*, que es el elemento que se va a representar, y debe devolver una cadena que represente el texto que va a mostrarse.

Ejemplo

En el ejemplo siguiente, la etiqueta muestra algunos detalles formateados de los elementos:

```
var my_list:mx.controls.List;

my_list.setSize(300, 100);

// Definir cómo aparecerán los datos de la lista.
my_list.labelFunction = function(item_obj:Object):String {
 var label_str:String = item_obj.label + " - Code is: " + item_obj.data;
 return label_str;
}

// Añadir datos a la lista.
my_list.addItem({data:"f", label:"Flash"});
my_list.addItem({data:"d", label:"Dreamweaver"});
my_list.addItem({data:"c", label:"ColdFusion"});
```

Véase también

`List.labelField`

List.length

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.length

Descripción

Propiedad (sólo lectura); número de elementos de la lista.

Ejemplo

En el siguiente ejemplo se muestra el número de elementos que actualmente están en el proveedor de datos de la lista:

```
var my_list:mx.controls.List;

// Añadir datos a la lista.
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var listLength_num:Number = my_list.length;
trace("Length of List: " + listLength_num);
```

List.maxHPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.maxHPosition

Descripción

Propiedad; especifica el número de píxeles que puede desplazarse la lista cuando `List.hScrollPolicy` está definida en `"on"`. La lista no mide con precisión la anchura del texto que contiene. Es preciso definir `maxHPosition` para indicar la cantidad de desplazamiento necesario. Si no se define la propiedad, la lista no se desplaza horizontalmente.

Ejemplo

En el ejemplo siguiente se crea una lista con 200 píxeles de desplazamiento horizontal:

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 200;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
```

Véase también

`List.hScrollPolicy`

List.multipleSelection

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.multipleSelection

Descripción

Propiedad; indica si se permite la selección múltiple (`true`) o única (`false`). El valor predeterminado es `false`.

Ejemplo

En el siguiente ejemplo se realiza una prueba para determinar si se pueden seleccionar varios elementos y, en caso afirmativo, se muestran las instrucciones en un componente Label. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. A continuación, arrastre un componente Label al escenario y asígnele el nombre de instancia **my_label**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.multipleSelection = true;

if (my_list.multipleSelection) {
  my_label.text = "Hold down Control or Shift to select multiple items";
  my_label.autoSize = "left";
}
```

List.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listInstance.removeAll()
```

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la lista.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

En el siguiente ejemplo se borran todos los elementos de un componente List cuando se hace clic en un botón. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. A continuación, arrastre un componente Button al escenario y asígnele el nombre de instancia **remove_button**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;
var remove_button:mx.controls.Button;

remove_button.label = "Remove";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  my_list.removeAll();
  evt_obj.target.enabled = false;
}
remove_button.addEventListener("click", buttonListener);
```

List.removeItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listInstance.removeItemAt(index)
```

Parámetros

index Número que indica la posición del elemento. El valor debe ser mayor que cero y menor que `List.length`.

Valor devuelto

Objeto; elemento eliminado (`undefined` si no hay elementos).

Descripción

Método; elimina el elemento de la posición de índice especificada. Los índices de la lista a partir del índice especificado se contraen una posición.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

En el siguiente ejemplo se borran los elementos seleccionados de un componente List cuando se hace clic en un botón. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. A continuación, arrastre un componente Button al escenario y asígnele el nombre de instancia **remove_button**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;
var remove_button:mx.controls.Button;

remove_button.label = "Remove";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
```

```
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  if (my_list.selectedIndex != undefined) {
    my_list.removeItemAt(my_list.selectedIndex);
  }
}
remove_button.addEventListener("click", buttonListener);
```

List.replaceItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.replaceItemAt(*index*, *label*[, *data*])

listInstance.replaceItemAt(*index*, *itemObject*)

Parámetros

index Número mayor que 0 y menor que `List.length` que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es optativo y puede ser de cualquier tipo.

itemObject Objeto que se utiliza como el elemento, que normalmente contiene las propiedades `label` y `data`.

Valor devuelto

Ninguno.

Descripción

Método; sustituye el contenido del elemento en el índice especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

En el siguiente ejemplo se reemplaza el elemento en la posición que está actualmente seleccionada. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. A continuación, arrastre un componente Button al escenario y asígnele el nombre de instancia **replace_button**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;
var replace_button:mx.controls.Button;

replace_button.label = "Replace";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  if (my_list.selectedIndex != undefined) {
    my_list.replaceItemAt(my_list.selectedIndex, {data:"flex",
  label:"Flex"});
  }
}
replace_button.addEventListener("click", buttonListener);
```

List.rowCount

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.rowCount

Descripción

Propiedad; número de filas que son al menos parcialmente visibles en la lista. Resulta útil cuando la escala de la lista está expresada en píxeles y es necesario contar las filas. Por el contrario, si se define el número de filas, se muestra un número exacto de filas sin que aparezca la fila final partida.

El código `my_list.rowCount = num` es equivalente al código `my_list.setSize(my_list.width, h)` (en el que `h` es la altura necesaria para mostrar el número `num` de elementos).

El valor predeterminado se basa en la altura de la lista definida durante la edición o con el método `List.setSize()` (véase `UIObject.setSize()`).

Ejemplo

En el ejemplo siguiente se revela el número de elementos visibles de una lista:

```
var rowCount = my_list.rowCount;
```

En el ejemplo siguiente, la lista muestra cuatro elementos:

```
my_list.rowCount = 4;
```

El siguiente ejemplo elimina la fila parcial del final de la lista, si la hay:

```
my_list.rowCount = my_list.rowCount;
```

El siguiente ejemplo define una lista con el número mínimo de filas que puede mostrar por completo:

```
my_list.rowCount = 1;
trace("my_list has " + my_list.rowCount + " rows");
```

El siguiente ejemplo cambia el tamaño de la lista mediante el método `setSize()` y, a continuación, establece un recuento de fila de 8 elementos:

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.setSize(200, 30);
my_list.rowCount = 8;
trace("my_list has " + my_list.rowCount + " rows.");
```

List.rowHeight

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.rowHeight

Descripción

Propiedad; altura de cada fila de la lista, expresada en píxeles. Las filas no aumentan de tamaño para que encaje la fuente configurada, por lo que definir la propiedad `rowHeight` es la mejor manera de asegurar que los elementos se muestren por completo. El valor predeterminado es 20.

Ejemplo

En el ejemplo siguiente se definen las filas en 30 píxeles:

```
my_list.rowHeight = 30;
```

En el siguiente ejemplo se establece la altura de fila en 30 píxeles y, a continuación, se cambia el tamaño de la lista para ajustarse al número total de elementos que contiene:

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.rowHeight = 30;
my_list.rowCount = my_list.length;
```

List.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // El código se escribe aquí.
};
listInstance.addEventListener("scroll", listenerObject);
```

Sintaxis 2:

```
on (scroll) {
  // El código se escribe aquí.
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, el evento `scroll` tiene una propiedad adicional, `direction`. Es una cadena con dos valores posibles, `"horizontal"` o `"vertical"`. En los eventos scroll de ComboBox, el valor es siempre `"vertical"`.

Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, `scroll`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher (API)" en la página 516.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de List. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `my_list` del componente List, envía "_level0.my_list" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se envía la dirección y la posición de la lista cada vez que se recorren los elementos de la lista:

```
var my_list:mx.controls.List;

my_list.rowCount = 2;
for (var i:Number = 0; i < 10; i++) {
  my_list.addItem({data:i, label:"Item #" + i});
}

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("list scrolled (direction:" + evt_obj.direction + ", position:" +
  evt_obj.position + ")");
};
my_list.addEventListener("scroll", listListener);
```

List.selectable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.selectable

Descripción

Propiedad; valor booleano que indica si la lista es seleccionable (`true`) o no (`false`). El valor predeterminado es `true`.

Ejemplo

En el siguiente ejemplo se impide que los usuarios puedan seleccionar elementos de la lista al establecer la propiedad seleccionable en `false`:

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.selectable = false;
```

List.selectedIndex

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedIndex

Descripción

Propiedad; índice seleccionado en una lista de selección única. El valor es `undefined` si no se selecciona nada; el valor es igual al último elemento seleccionado si se seleccionan varios. Si se asigna un valor a `selectedIndex`, se borra la selección actual y se selecciona el elemento indicado.

Si se utiliza la propiedad `selectedIndex` para cambiar la selección, no se envía un evento `change`. Para enviar el evento `change`, utilice el código siguiente:

```
my_list.dispatchEvent({type:"change", target:my_list});
```

Ejemplo

En el siguiente ejemplo se selecciona el primer elemento de una lista de forma predeterminada y se muestra el índice del elemento actualmente seleccionado siempre que el usuario elige un elemento nuevo:

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Seleccionar primer elemento de forma predeterminada.
my_list.selectedIndex = 0;

var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  trace("selectedIndex = " + evt_obj.target.selectedIndex);
}
my_list.addEventListener("change", listListener);
```

Véase también

`List.selectedIndices`, `List.selectedItem`, `List.selectedItems`

List.selectedIndices

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`listInstance``.selectedIndices`

Descripción

Propiedad; matriz de índices de los elementos seleccionados. Si se asigna esta propiedad, se sustituye la selección actual. Si `selectedIndices` se define en una matriz de longitud 0 (o `undefined`), se borra la selección actual. El valor es `undefined` si no se selecciona nada.

La propiedad `selectedIndices` muestra el orden en el que se han seleccionado los elementos. Si se hace clic en el segundo elemento, después en el tercero y después en el primero, `selectedIndices` devuelve `[1,2,0]`.

Ejemplo

En el ejemplo siguiente se recuperan los índices seleccionados:

```
var selIndices:Array = my_list.selectedIndices;
```

En el ejemplo siguiente se seleccionan cuatro elementos:

```
var my_array = new Array (1, 4, 5, 7);
my_list.selectedIndices = my_array;
```

En el siguiente ejemplo se seleccionan dos elementos de la lista de forma predeterminada y se muestra su propiedad label en el panel Salida:

```
my_list.multipleSelection = true;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.selectedIndices = [0, 2];

var numSelected:Number = my_list.selectedIndices.length;
for (var i:Number = 0; i < numSelected; i++) {
  trace("selectedIndices[" + i + "] = "+
  my_list.getItemAt(my_list.selectedIndices[i]).label);
}
```

Véase también

`List.selectedIndex`, `List.selectedItem`, `List.selectedItems`

List.selectedItem

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedItem

Descripción

Propiedad (sólo lectura); objeto de elemento de una lista de selección única. En una lista de selección múltiple con varios elementos seleccionados, `selectedItem` devuelve el elemento seleccionado en último lugar. Si no hay selección, el valor es `undefined`.

Ejemplo

En el siguiente ejemplo se muestra la etiqueta seleccionada:

```
trace(my_list.selectedItem.label);
```

En el siguiente ejemplo se muestra el contenido de un elemento seleccionado cuando el usuario selecciona un elemento nuevo de la lista:

```
my_list.multipleSelection = true;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Crear un objeto detector.
var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  // Mostrar cada propiedad del objeto.
  var tempStr:String = "[object";
  for (var i:String in evt_obj.target.selectedItem) {
    tempStr += " " + i + ":'" + evt_obj.target.selectedItem[i]+"'";
  }
  tempStr += "]";
  trace(tempStr);
};
// Añadir detector.
my_list.addEventListener("change", listListener);
```

Véase también

`List.selectedIndex`, `List.selectedIndices`, `List.selectedItems`

List.selectedItems

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedItems

Descripción

Propiedad (sólo lectura); matriz de objetos del elemento seleccionado. En una lista de selección múltiple, `selectedItems` permite acceder al conjunto de elementos seleccionados como objetos de elemento.

Ejemplo

En el ejemplo siguiente se recupera la matriz de objetos de elemento seleccionados:

```
var myObjArray:Array = my_list.selectedItems;
```

En el siguiente ejemplo se muestran dos instancias de List en el escenario. Cuando un usuario selecciona un elemento de la primera lista, el elemento seleccionado se copia en la segunda lista. Para probar este código, debe añadir una copia del componente List a la biblioteca del documento actual. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.createClassObject(mx.controls.List, "my_list", 10,
  {multipleSelection:true});
my_list.setSize(200, 100);

this.createClassObject(mx.controls.List, "selectedItems_list", 20,
  {selectable:false});
selectedItems_list.setSize(200, 100);
selectedItems_list.move(0, 110);

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  trace("You have selected " + my_list.selectedItems.length + " items.");
  selectedItems_list.dataProvider = my_list.selectedItems;
}
my_list.addEventListener("change", listListener);
```

Véase también

`List.selectedIndex`, `List.selectedItem`, `List.selectedIndices`

List.setPropertiesAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.setPropertiesAt(*index*, *styleObj*)

Parámetros

index Número mayor que 0 o menor que `List.length` que indica el índice del elemento que va a cambiarse.

styleObj Objeto que enumera las propiedades y valores que van a definirse.

Valor devuelto

Ninguno.

Descripción

Método; aplica las propiedades especificadas al elemento especificado. Las propiedades admitidas son `icon` y `backgroundColor`.

Ejemplo

En el siguiente ejemplo se cambia el color de fondo del tercer elemento a rojo y se le asigna un icono. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **my_list**. A continuación, añada un símbolo MovieClip/Graphic a la biblioteca con un identificador de vinculación de "file". Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.setPropertiesAt(2, {backgroundColor:0xFF0000, icon: "file"});
```

List.sortItems()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.sortItems(*compareFunc*)

Parámetros

compareFunc Referencia a una función. Esta función se utiliza para comparar dos elementos y determinar su orden de clasificación.

Para más información, consulte %{sort (método Array.sort)}% en *Referencia del lenguaje ActionScript 2.0*.

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos de la lista mediante la función especificada en el parámetro *compareFunc*.

Ejemplo

En el ejemplo siguiente se ordenan los elementos por etiquetas en mayúsculas. Observe que los parámetros `a` y `b` que pasan a la función son elementos que tienen las propiedades `label` y `data`.

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.sortItems(upperCaseFunc);
function upperCaseFunc(a:Object, b:Object):Boolean {
    return (a.label.toUpperCase() > b.label.toUpperCase());
}
```

Véase también

`List.sortItemsBy()`

List.sortItemsBy()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listInstance.sortItemsBy(fieldName, optionsFlag)
listInstance.sortItemsBy(fieldName, order)
```

Parámetros

fieldName Cadena que especifica el nombre del campo que se va a utilizar para la ordenación. Normalmente, este valor es `"label"` o `"data"`.

order Cadena que especifica si los elementos deben clasificarse en orden ascendente (`"ASC"`) o descendente (`"DESC"`).

optionsFlag Permite realizar varias ordenaciones de tipos distintos en una única matriz sin necesidad de duplicar toda la matriz ni de volver a ordenarla repetidas veces.

Los valores posibles de *optionsFlag* son:

- `Array.DESCENDING`, que ordena de mayor a menor.
- `Array.CASEINSENSITIVE`, que ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`, que ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas, que se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado esa etiqueta.
- `Array.UNIQUESORT`, que devuelve un código de error (0) en lugar de una matriz ordenada si dos objetos de la matriz son idénticos o lo son sus campos de ordenación.
- `Array.RETURNINDEXEDARRAY`, que devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, la siguiente matriz devolvería la segunda línea de código y la matriz no cambiaría:

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
my_array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valor devuelto

Ninguno.

Descripción

Método; ordena los elementos de la lista en el orden especificado mediante el nombre de campo especificado. Si los elementos de *fieldName* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *fieldName* suele ser `"label"` o `"data"`, pero puede especificarse cualquier valor de datos primitivos.

Esta es la forma más rápida para ordenar datos en un componente. También mantiene el estado de selección del componente. El método `sortItemsBy()` es rápido porque no ejecuta ningún código ActionScript mientras ordena. El método `sortItems()` necesita ejecutar una función de comparación de ActionScript y, por tanto, es más lento.

Ejemplo

En el código siguiente, los elementos de la lista se ordenan en sentido ascendente por sus etiquetas:

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.sortItemsBy("label", "ASC");
```

Véase también

`List.sortItems()`

List.vPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.vPosition

Descripción

Propiedad, define el elemento visible situado en la parte superior de la lista. Si establece esta propiedad en un número de índice que no existe, la lista se desplaza hasta el índice más cercano. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se muestra el valor actual de la vPosition de la lista cuando se recorre el contenido de la lista:

```
my_list.setSize(200, 60);
my_list.rowCount = 4;
my_list.vPosition = 2;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"flex", label:"Flex"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"fireworks", label:"Fireworks"});
my_list.addItem({data:"contribute", label:"Contribute"});
my_list.addItem({data:"breeze", label:"Breeze"});

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("my_list.vPosition = " + my_list.vPosition);
}
my_list.addEventListener("scroll", listListener);
```

List.vScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

listInstance.vScrollPolicy

Descripción

Propiedad; cadena que determina si la lista admite desplazamiento vertical. El valor de esta propiedad puede ser `"on"`, `"off"` o `"auto"`. El valor `"auto"` muestra la barra de desplazamiento cuando es necesario.

Ejemplo

En el siguiente ejemplo se desactiva la barra de desplazamiento vertical de una lista:

```
var my_list:mx.controls.List;

my_list.setSize(200, 60);
my_list.rowCount = 4;
my_list.vScrollPolicy = "off";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"flex", label:"Flex"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"fireworks", label:"Fireworks"});
my_list.addItem({data:"contribute", label:"Contribute"});
my_list.addItem({data:"breeze", label:"Breeze"});

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("my_list.vPosition = " + my_list.vPosition);
}
my_list.addEventListener("scroll", listListener);
```

Aún puede crear desplazamientos mediante `List.vPosition`, o utilizando el ratón o el teclado.

Véase también

`List.vPosition`

CAPÍTULO 28

Componente Loader

28

El componente Loader es un contenedor que puede mostrar archivos SWF o JPEG (pero no archivos JPEG *progresivos*). Se puede ajustar el tamaño del contenido del componente o cambiar su tamaño para que quepa el contenido. De forma predeterminada, el contenido se ajusta al componente Loader. También puede cargar el contenido en tiempo de ejecución y controlar el progreso de carga (aunque después de que el contenido se haya cargado, se almacena en la caché y el progreso salta al 100% rápidamente).

El componente Loader no se puede seleccionar. Sin embargo, sí se puede seleccionar el contenido cargado en el componente Loader e interactuar con él. Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia de Loader refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes.

Puede utilizar el panel Accesibilidad para que los lectores de pantalla tengan acceso al contenido del componente Loader. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Utilización del componente Loader

Puede utilizar el componente Loader siempre que necesite recuperar contenido de una ubicación remota y colocarlo en una aplicación de Flash. Por ejemplo, puede utilizar el componente Loader para añadir el logotipo de una empresa (archivo JPEG) a un formulario. También puede utilizar el componente para aprovechar trabajo de Flash que ya haya terminado. Por ejemplo, si ya ha creado una aplicación de Flash y desea ampliarla, puede utilizar el componente Loader para colocar la aplicación antigua en una nueva, tal vez como sección de una interfaz de fichas. En otro ejemplo, puede utilizar el componente Loader en una aplicación que muestra fotografías. Utilice `Loader.load()`, `Loader.percentLoaded` y `Loader.complete` para controlar el tiempo de carga de imágenes y mostrar al usuario barras de progreso durante la carga.

Si carga determinados componentes de la versión 2 de la arquitectura de componentes de Macromedia en un archivo SWF o en el componente Loader, es posible que los componentes no funcionen correctamente. Algunos de estos componentes son: Alert, ComboBox, DateField, Menu, MenuBar y Window.

Use la propiedad `_lockroot` al llamar a `loadMovie()` o al cargar en el componente Loader. Si está usando el componente Loader, añada el código siguiente:

```
myLoaderComponent.content._lockroot = true;
```

Si está usando un clip de película con una llamada a `loadMovie()`, añada el código siguiente:

```
myMovieClip._lockroot = true;
```

Si no establece `_lockroot` en `true` en el clip de película de cargador, el cargador sólo podrá acceder a su propia biblioteca, pero no a la biblioteca del clip de película cargado.

Flash Player 7 admite la propiedad `_lockroot`. Para más información sobre esta propiedad, consulte %{_lockroot (propiedad MovieClip._lockroot)}% en *Referencia del lenguaje ActionScript 2.0*.

Los componentes como Loader, ScrollPane y Window tienen eventos para determinar que ha finalizado la carga del contenido. De este modo, si desea establecer propiedades en el contenido de un componente Loader, ScrollPane o Window, añada la sentencia de propiedad en un controlador de eventos "complete", como se muestra en el siguiente ejemplo:

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._rotation= 45;
}
my_loader.addEventListener("complete", loadtest)
```

Para más información, consulte "Loader.complete" en la página 852.

Parámetros del componente Loader

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Loader en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

autoload indica si el contenido debe cargarse automáticamente (`true`) o si es preciso esperar hasta que se llame al método `Loader.load()` (`false`). El valor predeterminado es `true`.

contentPath URL absoluta o relativa que indica el archivo que debe cargarse en el cargador. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La dirección URL debe estar en el mismo subdominio que la URL donde reside actualmente el contenido de Flash. Tanto si van a utilizarse en Flash Player como en el modo de prueba, todos los archivos SWF deben almacenarse en la misma carpeta, y los nombres de archivo no deben incluir especificaciones de carpeta ni de unidad de disco. El valor predeterminado es `undefined` hasta que se inicia la carga.

El componente Loader puede cargar contenido de otros dominios *si* se dispone de archivos de política en tales dominios. Consulte "Permiso de carga de datos de varios dominios" en *Aprendizaje de ActionScript 2.0 en Flash*.

scaleContent indica si el contenido se redimensiona para adaptarse al componente Loader (`true`) o si el componente Loader se redimensiona para adaptarse al contenido (`false`). El valor predeterminado es `true`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente Loader en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Loader con sus métodos, propiedades y eventos. Para más información, consulte "Clase Loader" en la página 845.

Creación de aplicaciones con el componente Loader

El siguiente procedimiento explica cómo añadir un componente Loader a una aplicación durante la edición. En este ejemplo, el componente Loader carga un logotipo JPEG de una URL imaginaria.

Para crear una aplicación con el componente Loader:

1. Arrastre un componente Loader desde el panel Componentes al escenario.
2. En el inspector de propiedades, introduzca el nombre de instancia **flower**.
3. Seleccione el componente en el escenario y en el inspector de componentes, e introduzca **http://www.flash-mx.com/images/image1.jpg** para el parámetro contentPath.

Para crear una instancia del componente Loader mediante código ActionScript:

1. Arrastre un componente Loader desde el panel Componentes a la biblioteca.
2. Seleccione el primer fotograma de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
this.createClassObject(mx.controls.Loader, "my_loader", 1);
my_loader.contentPath = "http://www.flash-mx.com/images/image1.jpg";
```

   Este script utiliza el método "UIObject.createClassObject()" en la página 1404 para crear la instancia de Loader.
3. Seleccione Control > Probar película.

Personalización del componente Loader

El componente Loader puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`).

El comportamiento de ajuste de tamaño del componente Loader se controla con la propiedad `scaleContent`. Cuando `scaleContent` es `true`, el contenido se redimensiona para adaptarse a los límites del componente Loader (y vuelve a redimensionarse cuando se llama al método `UIObject.setSize()`). Cuando `scaleContent` es `false`, el tamaño del componente se ajusta al tamaño del contenido y el método `UIObject.setSize()` no tiene efecto alguno.

Utilización de estilos con el componente Loader

El componente Loader utiliza los siguientes estilos.

Estilo	Tema	Descripción
`borderStyle`	Ambos	El componente Loader utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es `"none"`.

Por ejemplo:

```
my_ldr.setStyle("backgroundColor", 0xEEEEEE);
```

Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Utilización de aspectos con el componente Loader

El componente Loader utiliza una instancia de RectBorder para definir el borde (véase "Clase RectBorder" en la página 1103).

Clase Loader

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > Loader

Nombre de clase de ActionScript mx.controls.Loader

Las propiedades de la clase Loader permiten definir el contenido que va a cargarse y controlar el progreso de carga en tiempo de ejecución.

Si una propiedad de la clase Loader se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.Loader.version);
```

NOTA El código `trace(myLoaderInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase Loader

En la tabla siguiente se muestra el método de la clase Loader.

Método	Descripción
`Loader.load()`	Carga el contenido especificado en la propiedad `contentPath`.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Loader de la clase UIObject. Al llamar a estos métodos desde el objeto Loader, debe utilizarse la forma *`LoaderInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Loader de la clase UIComponent. Al llamar a estos métodos desde el objeto Loader, debe utilizarse la forma *LoaderInstance.methodName*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase Loader

En la tabla siguiente se enumeran las propiedades de la clase Loader.

Propiedad	Descripción
`Loader.autoLoad`	Valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso llamar a `Loader.load()` (`false`).
`Loader.bytesLoaded`	Propiedad de sólo lectura que indica el número de bytes que se han cargado.
`Loader.bytesTotal`	Propiedad de sólo lectura que indica el número total de bytes del contenido.
`Loader.content`	Referencia al contenido del componente Loader. Es una propiedad de sólo lectura.
`Loader.contentPath`	Cadena que indica la URL del contenido que va a cargarse.
`Loader.percentLoaded`	Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.
`Loader.scaleContent`	Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader (`true`) o si el componente Loader se redimensiona para adaptarse al contenido (`false`).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Loader de la clase UIObject. Al acceder a estas propiedades desde el objeto Loader, debe utilizarse la forma *LoaderInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
UIObject.height	Altura del objeto, expresada en píxeles. Sólo lectura.
UIObject.left	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Loader de la clase UIComponent. Al acceder a estas propiedades desde el objeto Loader, debe utilizarse la forma *LoaderInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase Loader

En la tabla siguiente se enumeran los eventos de la clase Loader.

Evento	Descripción
`Loader.complete`	Se activa cuando el contenido termina de cargarse.
`Loader.progress`	Se activa mientras se carga el contenido.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Loader de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Loader de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Loader.autoLoad

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.autoLoad

Descripción

Propiedad; valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso esperar a que se llame a `Loader.load()` (`false`). El valor predeterminado es `true`.

Ejemplo

El código siguiente define que el componente Loader espere a que se llame a `Loader.load()`:

```
loader.autoload = false;
```

Loader.bytesLoaded

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.bytesLoaded

Descripción

Propiedad (sólo lectura); número de bytes del contenido cargado. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

Ejemplo

Con un componente Loader y un componente ProgressBar en la biblioteca del documento actual, el siguiente código crea instancias de ambos componentes. A continuación crea un objeto detector con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia `my_ldr`.

Cuando se crea una instancia con `createClassObject()`, es preciso colocarla en el escenario con `move()`. Véase `UIObject.move()`.

```
import mx.controls.Loader;
import mx.controls.ProgressBar;

System.security.allowDomain("http://www.flash-mx.com");

this.createClassObject(Loader, "my_ldr", 10);
this.createClassObject(ProgressBar, "my_pb", 20, {source:"my_ldr"});

my_ldr.move(1, 50);
my_pb.move(1, 1);

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object) {
 // evt_obj.target es el componente que genera el evento progress,
 // es decir, el componente Loader.
 my_pb.setProgress(my_ldr.bytesLoaded, my_ldr.bytesTotal);
 // Mostrar progreso.
};
my_ldr.addEventListener("progress", loaderListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Loader.bytesTotal

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`loaderInstance``.bytesTotal`

Descripción

Propiedad (sólo lectura); tamaño del contenido, expresado en bytes. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

Ejemplo

El código siguiente crea una barra de progreso y un componente Loader. A continuación crea un objeto detector de carga con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia de `my_ldr` de este modo:

```
import mx.controls.Loader;
import mx.controls.ProgressBar;
this.createClassObject(ProgressBar, "my_pb", 998);
this.createClassObject(Loader, "my_ldr", 999);
my_pb.move(1, 1);
my_ldr.move(1, 50);
my_pb.source = "my_ldr";
var loadListener:Object = new Object();
loadListener.progress = function(eventObj){
  // eventObj.target es el componente que genera el evento progress,
  // es decir, el componente Loader.
  my_pb.setProgress(my_ldr.bytesLoaded, my_ldr.bytesTotal); // Mostrar
  progreso.
}
my_ldr.addEventListener("progress", loadListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Véase también

`Loader.bytesLoaded`

Loader.complete

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObj:Object){
  // ...
};
loaderInstance.addEventListener("complete", listenerObject);
```

Sintaxis 2:

```
on (complete) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando finaliza la carga del contenido.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*loaderInstance*) distribuye un evento (en este caso, `complete`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de Loader. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myLoaderComponent` del componente Loader, envía "_level0.myLoaderComponent" al panel Salida:

```
on (complete) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se crea un componente Loader `my_ldr` y, a continuación, se establece un objeto detector para un evento `complete`. El ejemplo carga una imagen desde una página Web. Cuando finaliza la carga, el detector muestra un mensaje en el panel Salida.

Arrastre un componente Loader a la biblioteca, a continuación, añada el siguiente código al primer fotograma de la línea de tiempo.

```
/**
 Se requiere:
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.flash-mx.com");

//Crear instancia de Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
```

```
// Crear un objeto detector.
var loaderListener:Object = new Object();
loaderListener.complete = function(evt_obj:Object){
   trace("loading complete");
}

// Añadir detector.
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.flash-mx.com/images/image2.jpg");
```

Loader.content

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.content

Descripción

Propiedad (sólo lectura); referencia a una instancia del clip de película que incluye el contenido del archivo cargado. El valor es `undefined` hasta que comienza la carga. Establezca propiedades para el contenido de una función de controlador de eventos para el evento `Loader.complete`.

Ejemplo

El componente Loader tiene un evento "complete", por lo que puede asegurarse de que el contenido se ha cargado completamente antes de intentar acceder a las propiedades del contenido de Loader.

El siguiente ejemplo utiliza la propiedad `Loader.content` de una función de controlador de eventos para el evento complete. Arrastre un componente Loader desde el panel Componentes a la biblioteca del documento actual para que el componente aparezca en la biblioteca. A continuación, añada el siguiente código ActionScript al primer fotograma de la línea de tiempo principal:

```
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";
//Asignar una variable al contenido.
var content_mc:MovieClip = my_ldr.content;

var loadtest:Object = new Object();
```

```
loadtest.complete = function(){
//Definir propiedades para el contenido.
  content_mc._alpha = 50;
  content_mc._rotation= 45;
  trace(content_mc._width);
}
my_ldr.addEventListener("complete", loadtest);
```

Loader.contentPath

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.contentPath

Descripción

Propiedad; cadena que indica la URL absoluta o relativa del archivo que va a cargarse en el componente Loader. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La URL debe pertenecer al mismo subdominio que el archivo SWF que va a cargarse.

Si utiliza Flash Player o el modo de prueba en Flash, todos los archivos SWF deben almacenarse en la misma carpeta, y los nombres de archivo no pueden incluir información de carpeta ni de unidad de disco.

Ejemplo

En el ejemplo siguiente se indica a la instancia del componente Loader que muestre el contenido del archivo logo.swf:

```
flower.contentPath = "http://www.flash-mx.com/images/image1.jpg"
```

En el siguiente ejemplo se descarga contenido de Loader cuando se hace clic en la instancia de botón `my_btn`:

```
flower.contentPath = "http://www.flash-mx.com/images/image1.jpg"
function clicked(){
  flower.contentPath = "";
}
my_btn.addEventListener("click", clicked);
```

Loader.load()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.load([*path*])

Parámetros

path Parámetro opcional que especifica el valor de la propiedad contentPath antes de iniciarse la carga. Si no se especifica ningún valor, se utiliza el valor actual de contentPath.

Valor devuelto

Ninguno.

Descripción

Método; indica al componente Loader que comience a cargar su contenido.

Ejemplo

En el siguiente ejemplo se crea una instancia de Loader, my_ldr, y una instancia de Button y se establece la propiedad del cargador autoload en false, de modo que la carga no comienza hasta que se llama al método load(). A continuación, el ejemplo establece contentPath en la ubicación Web de una imagen y crea un detector para un evento click en el botón. Cuando el usuario hace clic en el botón, el controlador de evento llama a my_ldr.load() para cargar la imagen. El controlador de eventos también desactiva el botón.

Arrastre un componente Loader y un componente Button desde el panel Componentes a la biblioteca y, a continuación, añada el siguiente código al primer fotograma de la línea de tiempo.

```
/**
 Se requiere:
  - Componente Loader en la biblioteca
  - componente Button en la biblioteca
*/

System.security.allowDomain("http://www.flash-mx.com");

//Crear instancia de Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
this.createClassObject(mx.controls.Button, "load_button", 20, {label:"Load
  image"});
```

```
my_ldr.move(0, 30);

my_ldr.autoLoad = false;
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";

var loadListener:Object = new Object();
loadListener.click = function (evt_obj:Object) {
 my_ldr.load();
 load_button.enabled = false;
}
load_button.addEventListener("click", loadListener);
```

Loader.percentLoaded

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.percentLoaded

Descripción

Propiedad (sólo lectura); número que indica el porcentaje de contenido cargado. Normalmente, esta propiedad se utiliza para presentar el progreso en un formato de lectura fácil para el usuario. Utilice el siguiente código para redondear la cifra al número entero más cercano:

```
Math.round(bytesLoaded/bytesTotal*100))
```

Ejemplo

En el ejemplo siguiente se crea una instancia de Loader y, a continuación, un objeto detector con un controlador `progress` averigua el porcentaje cargado y lo envía al panel Salida:

```
import mx.controls.Loader;
this.createClassObject(Loader, "my_ldr", 999);
var loadListener:Object = new Object();
loadListener.progress = function(eventObj) {
  // eventObj.target es el componente que genera el evento progress,
  // es decir, el componente Loader.
  trace("The image is "+my_ldr.percentLoaded+"% loaded.");
  // Realizar seguimiento del progreso de carga.
};
my_ldr.addEventListener("progress", loadListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Loader.progress

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
loaderInstance.addEventListener("progress", listenerObject);
```

Sintaxis 2:

```
on (progress) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. Este evento se produce cuando la carga se activa con el parámetro `autoload` o con una llamada a `Loader.load()`. El evento de progreso no siempre se difunde y el evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede cuando el contenido cargado es un archivo local.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*`loaderInstance`*) distribuye un evento (en este caso, `progress`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de Loader. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myLoaderComponent` del componente Loader, envía "_level0.myLoaderComponent" al panel Salida:

```
on (progress) {
  trace(this);
}
```

Ejemplo

El código siguiente crea una instancia de Loader y después un objeto detector con un controlador de eventos para el evento `progress` que envía un mensaje al panel Salida con el porcentaje de contenido cargado:

```
//Crear instancia de Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);

// Crear un objeto detector.
var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object){
  // evt_obj.target es el componente que genera el evento progress,
  // es decir, el componente Loader.
  trace("image is " + my_ldr.percentLoaded + "% loaded.");
}

// Añadir detector.
my_ldr.addEventListener("progress", loaderListener);

//Asignar la ruta del contenido de Loader.
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";
```

Loader.scaleContent

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

loaderInstance.scaleContent

Descripción

Propiedad; indica si el contenido se redimensiona para adaptarse al componente Loader (true) o si el componente Loader se redimensiona para adaptarse al contenido (`false`). El valor predeterminado es `true`.

Ejemplo

El código siguiente indica al componente Loader que cambie de tamaño para ajustarse a su contenido:

```
my_ldr.scaleContent = false;
```

CAPÍTULO 29

Componentes multimedia (sólo en Flash Professional)

29

Los componentes de flujo de medios facilitan la incorporación de flujo de medios en las presentaciones de Macromedia Flash. Estos componentes permiten presentar los medios de varias maneras.

Puede utilizar los tres componentes multimedia siguientes:

- El componente MediaDisplay permite reproducir medios sin interrupción en el contenido de Flash sin necesidad de utilizar una interfaz de usuario como soporte. Este componente puede utilizarse con datos de vídeo y de audio. Al utilizar este componente por sí solo, el usuario no tiene control sobre los medios.
- El componente MediaController proporciona controles de interfaz de usuario estándar (reproducción, pausa, etc.) para la reproducción multimedia. Los medios no se cargan nunca en MediaController y éste tampoco los reproduce; dicho componente sólo se utiliza para controlar la reproducción en una instancia de MediaPlayback o MediaDisplay. El componente MediaController presenta un "cajón" que muestra el contenido de los controles de reproducción al colocar el ratón sobre el componente.
- El componente MediaPlayback es una combinación de los componentes MediaDisplay y MediaController; dicho componente proporciona métodos para reproducir sin interrupción el contenido de los medios.

Debe considerar los puntos siguientes, relacionados con los componentes multimedia:

- Los componentes multimedia requieren Flash Player 6 o posterior. En Flash Player 6, los componentes multimedia sólo admiten archivos FLV mediante Flash Communication Server y no mediante HTTP.
- Los componentes multimedia no admiten la función para explorar hacia adelante y hacia atrás. No obstante, puede conseguir esta función moviendo el deslizador del reproductor.
- Sólo se reflejan en la previsualización dinámica el tamaño del componente y el aspecto del controlador.
- Los componentes multimedia no admiten la accesibilidad.

Interacción con los componentes multimedia (sólo en Flash Professional)

Los componentes de flujo MediaPlayback y MediaController responden a las acciones realizadas con el ratón y el teclado; el componente MediaDisplay, no. En la tabla siguiente se resumen las acciones para los componentes MediaPlayback y MediaController después de seleccionarlos:

Destino	Navegación	Descripción
Controles de reproducción de un controlador determinado	Desplazamiento del ratón	Se resalta el botón.
Controles de reproducción de un controlador determinado	Hacer clic en el botón izquierdo del ratón	Los usuarios pueden hacer clic en los controles de reproducción para manipular la reproducción de los medios de audio y de vídeo. Los botones Pausar/Reproducir e Ir al principio/Ir al final se comportan como los botones estándar. Al presionar el botón del ratón, el botón de la pantalla se resalta en estado presionado y al soltar el botón del ratón, el botón de la pantalla vuelve a tener el aspecto de no estar seleccionado. El botón Ir al final se desactiva cuando se reproducen archivos de medios FLV.

Destino	Navegación	Descripción
Controles deslizantes de un controlador determinado	Mover deslizador hacia atrás y hacia adelante	La barra de reproducción indica la posición del usuario en el medio; el deslizador del reproductor se mueve horizontalmente (de forma predeterminada) para indicar la reproducción de principio (izquierda) a fin (derecha). El deslizador se mueve de abajo arriba cuando los controles están orientados verticalmente. A medida que el deslizador se mueve de izquierda a derecha, resalta el espacio de pantalla anterior para indicar que el contenido del mismo se ha vuelto a reproducir o seleccionar. El espacio de pantalla situado delante del deslizador permanece sin resaltar hasta que pasa el deslizador. Los usuarios pueden arrastrar el deslizador para actuar sobre la posición de la reproducción de los medios. Si se están reproduciendo los medios, la reproducción automática comienza en el punto en que se suelta el ratón. Si los medios están en pausa, el usuario puede mover y soltar el deslizador, y los medios permanecerán en posición de pausa. También hay un deslizador de volumen, que se mueve de izquierda (sin sonido) a derecha (volumen máximo) en el diseño horizontal y vertical.

Destino	Navegación	Descripción
Navegación del controlador de reproducción	Tab, Mayús+Tab	Cambia la selección de un botón a otro en el componente del controlador, donde se resalta el elemento seleccionado. Esta navegación funciona con los controles Pausar/ Reproducir, Ir al principio, Ir al final, Volumen desactivado y Volumen máx. La selección cambia de izquierda a derecha y de arriba abajo a medida que los usuarios se desplazan por los elementos con la tecla de tabulación. La combinación de Mayús+Tab mueve la selección de derecha a izquierda y de abajo arriba. Tras seleccionar un elemento mediante la tecla Tab, el control pasa inmediatamente al botón Reproducir/Pausar. Cuando se selecciona el botón Volumen máx. y se presiona la tecla Tab, la selección pasa al siguiente control en el índice de tabulación del escenario.
Botón de un control determinado	Barra espaciadora o Intro/Retorno	Marca el elemento seleccionado. Tras presionar la tecla adecuada, el botón aparece en estado presionado. Al soltarla, el botón vuelve a tener el estado de selección que se obtiene cuando se pasa el ratón por encima.

Aspectos básicos de los componentes multimedia (sólo en Flash Professional)

En esta sección se proporciona información general sobre el funcionamiento de los componentes multimedia. La mayor parte de las propiedades que figuran en esta sección pueden configurarse con el inspector de componentes. (Véase "Utilización del inspector de componentes con componentes multimedia" en la página 872.)

Aparte de las propiedades de diseño descritas más adelante en esta sección, para los componentes MediaDisplay y MediaPlayback pueden establecerse las propiedades siguientes:

- El tipo de medios, que puede establecerse en MP3 o FLV (véase `Media.mediaType` y `Media.setMedia()`).
- La ruta del contenido relativo o absoluto, que aloja el archivo de medios que se va a reproducir sin interrupción (véase `Media.contentPath`).
- Objetos de cuepoint, junto con el nombre correspondiente, la hora y las propiedades del reproductor (véase `Media.addCuePoint()` y `Media.cuePoints`). El nombre del cuepoint es arbitrario; asigne un nombre que tenga un significado al utilizar eventos de detector y de trazado. Un cuepoint difunde un evento `cuePoint` cuando el valor de su propiedad de tiempo es igual al de la ubicación de la cabeza lectora del componente MediaPlayback o MediaDisplay con el que está asociado. La propiedad del reproductor es una referencia a la instancia de MediaPlayback con la que está asociada. Aun así, puede eliminar cuepoints con `Media.removeCuePoint()` y `Media.removeAllCuePoints()`.

Los componentes de flujo de medios difunden varios eventos relacionados. Para poner en movimiento otros elementos pueden utilizarse los eventos de difusión siguientes:

- Un evento `change` se difunde continuamente mediante los componentes MediaDisplay y MediaPlayback mientras se reproducen los medios. (Véase `Media.change`.)
- Un evento `progress` se difunde continuamente mediante los componentes MediaDisplay y MediaPlayback mientras se cargan los medios. (Véase `Media.progress`.)
- Un evento `click` se difunde mediante los componentes MediaController y MediaPlayback cuando el usuario hace clic en el botón Pausar/Reproducir. En este caso, la propiedad `detail` del objeto del evento proporciona información acerca del botón en el que se ha hecho clic. (Véase `Media.click`.)
- Un evento `volume` se difunde mediante los componentes MediaController y MediaPlayback cuando el usuario ajusta los controles de volumen. (Véase `Media.volume`.)
- Un evento `playheadChange` se difunde mediante los componentes MediaController y MediaPlayback cuando el usuario mueve el deslizador del reproductor o hace clic en el botón Ir al principio o Ir al final. (Véase `Media.playheadChange`.)

El componente MediaDisplay funciona con el componente MediaController. Al estar combinados, los componentes se comportan de una manera similar al componente MediaPlayback, pero ofrecen más flexibilidad con respecto al aspecto y funcionamiento de la presentación de los medios.

Aspectos básicos del componente MediaDisplay

Al colocar un componente MediaDisplay en el escenario, no tiene interfaz de usuario. Se trata simplemente de un recuadro en el que albergar y reproducir los medios. Las propiedades siguientes afectan al aspecto de los medios de vídeo que se reproducen en un componente MediaDisplay:

- `Media.aspectRatio`
- `Media.autoSize`
- Altura (en el inspector de propiedades)
- Anchura (en el inspector de propiedades)

NOTA	El usuario no verá nada, a menos que se reproduzca algún medio.

La propiedad `Media.aspectRatio` tiene prioridad sobre las otras propiedades. Si `Media.aspectRatio` se establece en `true` (valor predeterminado), el componente siempre vuelve a ajustar el medio que se reproduce para mantener la proporción del medio.

Para los archivos FLV, si `Media.autoSize` se establece en `true`, el medio se reproduce con su tamaño preferido, independientemente del tamaño del componente. Esto implica que si el tamaño de la instancia de MediaDisplay es diferente al tamaño de los medios, éstos rebasarán los límites de la instancia o no llenarán el tamaño de la misma. Si `Media.autoSize` se establece en `false`, Flash utiliza el tamaño de la instancia mientras sea posible, al tiempo que respeta la proporción. Si `Media.autoSize` y `Media.aspectRatio` se establecen en `false`, se utiliza el tamaño exacto del componente.

NOTA	Como no hay ninguna imagen que pueda mostrarse con los archivos MP3, establecer `Media.autoSize` no tendrá ningún efecto. Para los archivos MP3, el tamaño utilizable mínimo es 60 píxeles de altura por 256 píxeles de anchura en la orientación predeterminada.

El componente MediaDisplay también admite la propiedad `Media.volume`. Esta propiedad acepta valores enteros de 0 (sin sonido) a 100 (volumen máximo). El valor predeterminado es 75.

Aspectos básicos del componente MediaController

La interfaz para el componente MediaController depende de las propiedades `Media.controllerPolicy` y `Media.backgroundStyle` correspondientes. La propiedad `Media.controllerPolicy` determina si el juego de controles de los medios está siempre visible, contraído o sólo se puede ver al colocar el puntero del ratón sobre la parte de control del componente. Al contraerse, el controlador dibuja una barra de progreso modificada, que es una combinación de la barra de carga y la barra de reproducción. La barra de progreso muestra el progreso de los bytes que se están cargando en la parte inferior de la barra y el progreso de la cabeza lectora justo encima. El controlador, cuando se expande, dibuja una versión ampliada de la barra de reproducción/barra de carga, que contiene los elementos siguientes:

- Etiquetas de texto a la izquierda, que indican el estado de reproducción (sin interrupción o pausada) y a la derecha, que indican la ubicación de la cabeza lectora (en segundos).
- Un indicador de la ubicación de la cabeza lectora.
- Un deslizador, que los usuarios pueden arrastrar para desplazarse por los medios.

El componente MediaController también proporciona los elementos siguientes:

- Un botón Pausar/Reproducir
- Los botones Ir al principio e Ir al final, que sirven para desplazarse al principio y al final de los medios, respectivamente.
- Un control de volumen, que consta de un deslizador, un botón para quitar el sonido y otro botón de volumen máximo.

Tanto el estado contraído como el estado expandido del componente MediaController utilizan la propiedad `Media.backgroundStyle`. Esta propiedad determina si el controlador dibuja un fondo cromático (valor predeterminado) o permite que el fondo del medio se visualice desde detrás de los controles.

El componente MediaController tiene una configuración de orientación, `Media.horizontal`, que puede utilizarse para dibujar el componente con la orientación horizontal (valor predeterminado) o vertical. Con la orientación horizontal, la barra de reproducción hace un seguimiento de los medios que se están reproduciendo de izquierda a derecha. Con la orientación vertical, la barra de reproducción hace un seguimiento de los medios de abajo arriba.

Los componentes MediaDisplay y MediaController pueden asociarse entre sí mediante los métodos `Media.associateDisplay()` y `Media.associateController()`. Estos métodos permiten a la instancia de MediaController actualizar sus controles en función de los eventos difundidos desde la instancia de MediaDisplay. Asimismo, permiten que el componente MediaDisplay reaccione a la configuración del usuario de MediaController.

Aspectos básicos del componente MediaPlayback

MediaPlayback contiene los subcomponentes MediaController y MediaDisplay. La escala de MediaController y MediaDisplay siempre se ajusta al tamaño total de la instancia de MediaPlayback.

El componente MediaPlayback utiliza `Media.controlPlacement` para determinar el diseño de los controles. Al establecer esta propiedad en `top`, `bottom`, `left` o `right`, se puede indicar dónde se dibujan los controles en relación con la pantalla. Por ejemplo, el valor `right` proporciona la orientación vertical a un control y lo coloca a la derecha de la pantalla.

Utilización de componentes multimedia (sólo en Flash Professional)

Con el rápido aumento de la utilización de los elementos multimedia, destinados a proporcionar información a los usuarios de Internet, muchos desarrolladores fomentan que los usuarios puedan reproducir sin interrupción los medios y controlarlos. Los componentes multimedia podrían utilizarse en los tipos de situaciones siguientes:

- Mostrar los medios que presentan a una empresa
- Reproducir sin interrupción películas o vistas previas de películas
- Reproducir sin interrupción canciones o fragmentos de canciones
- Proporcionar material de aprendizaje multimedia

Utilización del componente MediaPlayback

Suponga que debe desarrollar un sitio Web que permita a los usuarios obtener una vista previa de los DVD y CD puestos a la venta en un entorno de medios de gran elaboración. El ejemplo siguiente muestra los pasos requeridos (se presupone que el sitio Web está preparado para poder insertar componentes de reproducción sin interrupción).

Para crear un documento de Flash que muestre una vista previa de un CD o DVD:

1. Seleccione Archivo > Nuevo y seleccione un documento de Flash.
2. Abra el panel Componentes y haga doble clic en el componente MediaPlayback para colocar una instancia de éste en el escenario.
3. Seleccione la instancia de componente MediaPlayback e introduzca el nombre de instancia **myMedia** en el inspector de propiedades.

4. En el inspector de componentes, defina el tipo de medio que se reproducirá sin interrupción (MP3 o FLV).
5. Si ha seleccionado FLV, introduzca la duración del vídeo en los cuadros de texto Video Length; utilice el formato *HH:MM:SS*.
6. Introduzca la ubicación del vídeo de vista previa en el cuadro de texto URL. Por ejemplo, podría introducir**www.helpexamples.com/flash/video/clouds.flv**.
7. Establezca las opciones que desee en las casillas de verificación Automatically Play, Use Preferred Media Size y Respect Aspect Ratio.
8. Establezca la colocación del control en el lado del componente MediaPlayback que desee.
9. Añada un cuepoint casi al final del medio haciendo clic en el botón Añadir (+); dicho cuepoint se utilizará con un detector para abrir una ventana emergente que anuncie que la película está a la venta. Asigne al cuepoint el nombre **cuePointName** y una posición cerca del final de la duración del medio.
10. Arrastre un componente Window desde el panel Componentes a la biblioteca del documento actual.

    Se colocará un símbolo denominado Window en la biblioteca y el componente Window estará disponible para el archivo SWF en tiempo de ejecución.
11. Cree un cuadro de texto y escriba un texto para informar al usuario de que la película está a la venta.
12. Seleccione Modificar > Convertir en símbolo para convertir el cuadro de texto en un clip de película y asígnele el nombre **mySale_mc**.
13. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el clip de película `mySale_mc` situado en la biblioteca; seleccione Vinculación y, a continuación, Exportar para ActionScript.

    Esto coloca el clip de película en la biblioteca de tiempo de ejecución.
14. Añada el siguiente código ActionScript al fotograma 1. Con este código se crea un detector para abrir una ventana emergente que informa al usuario de que la película está a la venta.

```
// Importar las clases necesarias para crear la ventana emergente
  dinámicamente.

import mx.containers.Window;
import mx.managers.PopUpManager;
```

```
// Crear un objeto detector para abrir la ventana emergente de venta.
var saleListener:Object = new Object();

saleListener.cuePoint = function(eventObj:Object) {

var saleWin:MovieClip = PopUpManager.createPopUp(_root, Window, false,
  {closeButton:true, title:"Movie Sale", contentPath:"mySale_mc"});

// Ampliar la ventana para que quepa el contenido.

saleWin.setSize(80, 80);
var delSaleWin:Object = new Object();
delSaleWin.click = function(eventObj:Object) {
  saleWin.deletePopUp();
}
saleWin.addEventListener("click", delSaleWin);

}

myMedia.addEventListener("cuePoint", saleListener);
```

15. Seleccione Control > Probar película para probar el archivo SWF.

    Cuando la aplicación llegue al tiempo de reproducción del cuepoint cuePointName, aparecerá una ventana emergente donde se mostrará el mensaje.

Utilización de los componentes MediaDisplay y MediaController

Si desea tener un mayor control sobre el aspecto y funcionamiento de la visualización de los medios, puede utilizar los componentes MediaDisplay y MediaController de forma conjunta. El ejemplo siguiente crea una aplicación Flash que muestra vistas previas de CD y DVD.

Para crear un documento de Flash que muestre una vista previa de un CD o DVD:

1. En Flash, seleccione Archivo > Nuevo; a continuación, seleccione Documento de Flash.
2. En el panel Componentes, arrastre un componente MediaController *y* un componente MediaDisplay al escenario.
3. Seleccione la instancia de MediaDisplay e introduzca el nombre de instancia **myDisplay** en el inspector de propiedades.
4. Seleccione la instancia de MediaController e introduzca el nombre de instancia **myController** en el inspector de propiedades.
5. Seleccione la instancia de MediaDisplay y abra la ficha Parámetros del inspector de componentes. Defina el tipo de medio que se reproducirá sin interrupción (MP3 o FLV).

6. Si ha seleccionado FLV, introduzca la duración del vídeo en los cuadros de texto Video Length mediante el formato *HH:MM:SS*.
7. Introduzca la ubicación del vídeo de vista previa en el cuadro de texto URL. Por ejemplo, podría introducir**www.helpexamples.com/flash/video/clouds.flv**.
8. Establezca las opciones que desee en las casillas de verificación Automatically Play, Use Preferred Media Size y Respect Aspect Ratio.
9. Seleccione la instancia de MediaController y, en la ficha Parámetros del inspector de componentes, defina la orientación como vertical estableciendo la propiedad `horizontal` en `false`.
10. En la ficha Parámetros del inspector de componentes, establezca `none` como valor de `backgroundStyle`.

    Esta acción especifica que la instancia de MediaController no debe dibujar un fondo, sino llenar los medios entre los controles.

    A continuación, utilice un comportamiento para asociar las instancias de MediaController y MediaDisplay de manera que la instancia de MediaController refleje de forma precisa el movimiento de la cabeza lectora y otras configuraciones de la instancia de MediaDisplay, y ésta responda a los clics de los usuarios.
11. Con la instancia de MediaController todavía seleccionada, abra el panel Comportamientos (Ventana > Comportamientos).
12. En el panel Comportamientos, haga clic en el botón Añadir (+) y seleccione Medios > Visualización asociada.
13. En la ventana Visualización asociada, seleccione `myDisplay` en `_root` y haga clic en Aceptar.

Para más información sobre cómo utilizar comportamientos con componentes multimedia, consulte "Control de componentes multimedia mediante comportamientos" en la página 873.

Utilización del inspector de componentes con componentes multimedia

El inspector de componentes facilita la definición, entre otros, de los parámetros y las propiedades de los componentes multimedia. Para utilizar este panel, haga clic en el componente del escenario que desee y, con el inspector de propiedades abierto, haga clic en Iniciar inspector de componentes. El inspector de componentes sirve para realizar las siguientes operaciones:

- Reproducir automáticamente el medio (véanse `Media.activePlayControl` y `Media.autoPlay`)
- Mantener o ignorar la proporción del elemento multimedia (véase `Media.aspectRatio`)
- Determinar si se cambiará automáticamente el tamaño del medio para adaptarlo a la instancia del componente (véase `Media.autoSize`)
- Activar o desactivar el fondo cromático (véase `Media.backgroundStyle`)
- Especificar la ruta del medio con formato de URL (véase `Media.contentPath`)
- Especificar la visibilidad de los controles de reproducción (véase `Media.controllerPolicy`)
- Añadir objetos cuepoint (véase `Media.addCuePoint()`)
- Eliminar objetos cuepoint (véase `Media.removeCuePoint()`)
- Establecer la orientación de las instancias de MediaController (véase `Media.horizontal`)
- Establecer el tipo de medio que se va a reproducir (véase `Media.setMedia()`)
- Establecer el tiempo de reproducción del medio FLV (véase `Media.totalTime`)
- Establecer los últimos dígitos de la visualización de tiempo para indicar los milisegundos o los fotogramas por segundo (fps)

Es importante que comprenda algunos conceptos del inspector de componentes para trabajar con él:

- El control de tiempo del vídeo no está disponible al seleccionar un tipo de vídeo MP3, ya que esta información se lee automáticamente cuando se utilizan archivos MP3. En el caso de los archivos FLV creados con Flash Video Exporter 1.0, debe introducir el tiempo total del medio (`Media.totalTime`) para que la barra de reproducción del componente MediaPlayback (o cualquier otro componente MediaController de sonido) pueda reflejar de forma precisa el progreso de la reproducción. Los archivos FLV creados con Flash Video Exporter 1.1 o posterior establecen la duración automáticamente.

- Si se selecciona el tipo de archivo FLV, aparecerán una opción de milisegundos y, si la opción de milisegundos no está seleccionada, un menú emergente de control de fotogramas por segundo (FPS). Si se selecciona la opción Milliseconds, el control FPS no aparece. En este modo, el tiempo que aparece en la barra de reproducción en tiempo de ejecución tiene el formato *HH:MM:SS.mmm* (*H* = horas, *M* = minutos, *S* = segundos, *m* = milisegundos), y se utiliza el valor de segundos para los cuepoints. Si la opción de milisegundos no está seleccionada, el control FPS está activado y el tiempo de la barra de reproducción tiene el formato *HH:MM:SS.FF* (*F* = fotogramas por segundo). Para los cuepoints, se utiliza el valor de fotogramas.

NOTA	El valor de fps sólo puede establecerse mediante el inspector de componentes. Si se utiliza ActionScript para definir un valor de fps, éste no tiene ningún efecto y se omite.

Control de componentes multimedia mediante comportamientos

Los comportamientos son scripts predefinidos de ActionScript que se añaden a una instancia, como un componente MediaDisplay, para controlar dicho objeto. Los comportamientos permiten añadir potencia, control y flexibilidad de codificación ActionScript al documento sin que tenga que crear el código ActionScript por sí mismo.

Para controlar un componente multimedia con un comportamiento, debe utilizar el panel Comportamientos a fin de aplicar dicho comportamiento a una instancia de componente multimedia determinada. Para ello, debe especificar el evento que activará el comportamiento (como por ejemplo, alcanzar un cuepoint específico), seleccionar un objeto de destino (los componentes multimedia que resultarán afectados por el comportamiento) y, si es necesario, seleccionar valores de configuración para el comportamiento (como el clip de película del medio al que desplazarse).

Con Flash Professional 8 se entregan los siguientes comportamientos que sirven para controlar componentes multimedia incorporados.

Comportamiento	Propósito	Parámetros
Controlador asociado	Asocia un componente MediaController con un componente MediaDisplay	Nombre de instancia de los componentes MediaController de destino
Visualización asociada	Asocia un componente MediaDisplay con un componente MediaController	Nombre de instancia de los componentes MediaController de destino

Comportamiento	Propósito	Parámetros
Navegación por cuepoints en fotogramas con nombre	Coloca una acción en una instancia de MediaDisplay o MediaPlayback que indica a un clip de película determinado que se desplace a un fotograma cuyo nombre sea igual al de un cuepoint especificado	Nombre de fotograma y nombre de cuepoint (los nombres deben ser idénticos)
Navegación por cuepoints en diapositivas	Desplaza un documento de Flash de diapositiva a una diapositiva cuyo nombre sea igual al de un cuepoint especificado	Nombre de diapositiva y nombre de cuepoint (los nombres deben ser idénticos)

Para asociar un componente MediaDisplay con un componente MediaController:

1. Coloque una instancia de MediaDisplay y una instancia de MediaController en el escenario.
2. Seleccione la instancia de MediaDisplay e introduzca el nombre de instancia **myMediaDisplay** con el inspector de propiedades.
3. Seleccione la instancia de MediaController que activará el comportamiento.
4. En el panel Comportamientos, haga clic en el botón Añadir (+) y seleccione Medios > Visualización asociada.
5. En la ventana Visualización asociada, seleccione `myMediaDisplay` en `_root` y haga clic en Aceptar.

NOTA	Si ha asociado el componente MediaDisplay con el componente MediaController, no es necesario que asocie el componente MediaController con el componente MediaDisplay.

Para asociar el componente MediaController con el componente MediaDisplay:

1. Coloque una instancia de MediaDisplay y una instancia de MediaController en el escenario.
2. Seleccione la instancia de MediaController e introduzca el nombre de instancia **myMediaController** con el inspector de propiedades.
3. Seleccione la instancia de MediaDisplay que activará el comportamiento.
4. En el panel Comportamientos, haga clic en el botón Añadir (+) y seleccione Medios > Controlador asociado.
5. En la ventana Controlador asociado, seleccione `myMediaController` en `_root` y haga clic en Aceptar.

Para utilizar un comportamiento Navegación por cuepoints en fotogramas con nombre:

1. Coloque una instancia de componente MediaDisplay o MediaPlayback en el escenario.
2. Seleccione el fotograma al que se desplazará el medio e introduzca el nombre de fotograma **myLabeledFrame** con el inspector de propiedades.
3. Seleccione la instancia de MediaDisplay o de MediaPlayback.
4. En el inspector de componentes, haga clic en el botón Añadir (+) e introduzca el tiempo de cuepoint en el formato *HH:MM:SS:mmm* o *HH:MM:SS:FF* y asigne el nombre **myLabeledFrame** al cuepoint.

   El cuepoint indica el tiempo que debe transcurrir antes de desplazarse al fotograma seleccionado. Por ejemplo, si desea saltar al fotograma `myLabeledFrame` del medio al cabo de 5 segundos, introduzca **5** en el cuadro de texto de los segundos y **myLabeledFrame** en el cuadro de texto del nombre.
5. En el panel Comportamientos, haga clic en el botón Añadir (+) y seleccione Medios > Navegación por cuepoints en fotogramas con nombre.
6. En la ventana Navegación por cuepoints en fotogramas con nombre, seleccione el clip `_root` y haga clic en Aceptar.

Para utilizar un comportamiento Navegación por cuepoints en diapositivas:

1. Abra el nuevo documento como presentación de diapositivas de Flash.
2. Coloque una instancia de componente MediaDisplay o MediaPlayback en el escenario.
3. En el panel Contorno de pantalla situado a la izquierda del escenario, haga clic en el botón Insertar pantalla (+) para añadir una segunda diapositiva; a continuación, seleccione la segunda diapositiva y asígnele el nombre **mySlide**.
4. Seleccione la instancia de MediaDisplay o de MediaController.
5. En el inspector de componentes, haga clic en el botón Añadir (+) e introduzca el tiempo de cuepoint en el formato *HH:MM:SS:mmm* o *HH:MM:SS:FF* y asigne el nombre **MySlide** al cuepoint.

   El cuepoint indica el tiempo que debe transcurrir antes de desplazarse a la diapositiva seleccionada. Por ejemplo, si desea saltar a la diapositiva `mySlide` del medio al cabo de 5 segundos, introduzca **5** en el cuadro de texto de los segundos y **mySlide** en el cuadro de texto de nombre.
6. En el panel Comportamientos, haga clic en el botón Añadir (+) y seleccione Medios > Navegación por cuepoints en diapositivas.
7. En la ventana Navegación por cuepoints en diapositivas, seleccione `Presentación` en el clip `_root` y haga clic en Aceptar.

Parámetros de los componentes multimedia (sólo en Flash Professional)

En las tablas siguientes se enumeran parámetros MediaDisplay, MediaController y MediaPlayback de edición que pueden establecerse para una instancia de componente multimedia determinada en el inspector de propiedades.

Parámetros de MediaDisplay

Nombre	Tipo	Valor predeterminado	Descripción
Automatically Play (`Media.autoPlay`)	Boolean	Selected	Determina si el medio se reproduce en cuanto se carga.
Use Preferred Media Size (`Media.autoSize`)	Boolean	Selected	Determina si el medio asociado con la instancia de MediaDisplay se adapta al tamaño del componente o simplemente utiliza su tamaño predeterminado.
FPS	Integer	30	Indica el número de fotogramas por segundo. Si la opción Milliseconds está seleccionada, este control está desactivado.
Cue Points (`Media.cuePoints`)	Array	Undefined	Matriz de objetos de cuepoint, cada uno con un nombre y una posición en el tiempo en un formato *HH:MM:SS:FF* válido (opción Milliseconds seleccionada) o *HH:MM:SS:mmm*.
FLV o MP3 (`Media.mediaType`)	FLV o MP3	`FLV`	Indica el tipo de medio que se va a reproducir.
Milliseconds	Boolean	Unselected	Determina si la barra de reproducción utiliza fotogramas o milisegundos, y si los cuepoints utilizan segundos o fotogramas. Si se selecciona esta opción, el control FPS no aparece.

Nombre	Tipo	Valor predeterminado	Descripción
URL (`Media.contentPath`)	String	Undefined	Cadena que contiene la ruta y el nombre de archivo del medio que va a reproducirse.
Video Length (`Media.totalTime`)	Integer	Undefined	Tiempo total necesario para reproducir el medio FLV. Esta configuración es necesaria para que la barra de reproducción funcione correctamente. Este control sólo es visible cuando el valor del tipo de medio es FLV.

Parámetros de MediaController

Nombre	Tipo	Valor predeterminado	Descripción
activePlayControl (`Media.activePlayControl`)	String: `pause` o `play`	`pause`	Determina si la barra de reproducción está en modo de reproducción o de pausa después de la creación de instancias. Este modo determina la imagen que se muestra en el botón Reproducir/Pausar, que es lo contrario del estado de reproducción/pausa en el que se encuentra realmente el controlador.
backgroundStyle (`Media.backgroundStyle`)	String: `default` o `none`	`default`	Determina si se extraerá el fondo cromático para la instancia de MediaController.
controllerPolicy (`Media.controllerPolicy`)	String: `auto`, `on` u `off`	`auto`	Determina si el controlador se abre o se cierra según la posición del ratón, o si se bloquea una vez que está abierto o cerrado.

Nombre	Tipo	Valor predeterminado	Descripción
horizontal (`Media.horizontal`)	Boolean	`true`	Determina si la parte de controlador de la instancia está orientada vertical u horizontalmente. El valor `true` indica que el componente estará orientado horizontalmente.
enabled	Boolean	`true`	Determina si el usuario puede modificar este control. El valor `true` indica que el control puede ser modificado.
visible	Boolean	`true`	Determina si el usuario puede visualizar este control. El valor `true` indica que el control está visible.

Parámetros de MediaPlayback

Nombre	Tipo	Valor predeterminado	Descripción
Control Placement (`Media.controlPlacement`)	String: `top`, `bottom`, `left`, `right`	`bottom`	Posición del controlador. El valor está relacionado con la orientación.
Control Visibility (`Media.controllerPolicy`)	Boolean	true	Determina si el controlador se abre o se cierra en función de la posición del ratón.
Automatically Play (`Media.autoPlay`)	Boolean	true	Determina si el medio se reproduce en cuanto se carga.
Use Preferred Media Size (`Media.autoSize`)	Boolean	true	Determina si la instancia de MediaController cambia de tamaño para adaptase al medio o utiliza otra configuración.
FPS	Integer	30	Número de fotogramas por segundo. Si la opción Milliseconds está seleccionada, este control está desactivado.

Nombre	Tipo	Valor predeterminado	Descripción
Cue Points (`Media.cuePoints`)	Array	undefined	Matriz de objetos de cuepoint, cada uno con un nombre y una posición en el tiempo en un formato *HH:MM:SS:mmm* válido (opción Milliseconds seleccionada) o *HH:MM:SS:FF*.
FLV o MP3 (`Media.mediaType`)	String: FLV o MP3	`FLV`	Indica el tipo de medio que se va a reproducir.
Milliseconds	Boolean	false	Determina si la barra de reproducción utiliza fotogramas o milisegundos, y si los cuepoints utilizan segundos o fotogramas. Cuando esta opción está seleccionada, el control FPS está desactivado.
URL (`Media.contentPath`)	String	undefined	Cadena que contiene la ruta y el nombre de archivo del medio que va a reproducirse.
Video Length (`Media.totalTime`)	Integer	undefined	Tiempo total necesario para reproducir el medio FLV. Esta configuración es necesaria para que la barra de reproducción funcione correctamente.

Creación de aplicaciones con componentes multimedia (sólo en Flash Professional)

La creación de contenido de Flash mediante componentes multimedia es bastante fácil y suele requerir pocos pasos. Este ejemplo muestra la manera de crear una aplicación para reproducir un pequeño archivo multimedia disponible públicamente.

Para añadir un componente multimedia a una aplicación:

1. En Flash, seleccione Archivo > Nuevo; a continuación, seleccione Documento de Flash.
2. En el panel Componentes, haga doble clic en el componente MediaPlayback para añadirlo al escenario.
3. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **myMedia**.
   - Haga clic en Iniciar inspector de componentes
4. En el inspector de componentes, introduzca**http://www.helpexamples.com/flash/video/water.flv** en el cuadro de texto URL.
5. Seleccione Control > Probar película, para ver la reproducción del medio.

Personalización de los componentes multimedia (sólo en Flash Professional)

Si desea cambiar la apariencia de los componentes multimedia, puede aplicarles aspectos. Para obtener una guía completa sobre la personalización de componentes, consulte Capítulo 5, “Personalización de componentes” en *Utilización de componentes*.

Utilización de estilos con componentes multimedia

Los componentes multimedia no admiten estilos.

Utilización de aspectos con componentes multimedia

Aunque los componentes multimedia no admiten la aplicación dinámica de aspectos, puede abrir el documento de origen de los componentes multimedia y cambiar sus activos para conseguir el aspecto deseado. Es aconsejable realizar una copia de este archivo y trabajar desde ella; de este modo, siempre tendrá el archivo de origen instalado, al cual podrá recurrir en caso necesario. Puede encontrar el documento de origen del componente multimedia en las ubicaciones siguientes:

- Windows: C:\Archivos de programa\Macromedia\Flash 8\<*idioma*>\Configuration\ComponentFLA\MediaComponents.fla
- Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/MediaComponents.fla

Clase Media (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > Media

Nombres de clase de ActionScript mx.controls.MediaController, mx.controls.MediaDisplay, mx.controls.MediaPlayback

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles para la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.MediaPlayback.version);
```

NOTA	El código `trace(myMediaInstance.version);` devuelve `undefined`.

Resumen de métodos para la clase Media

En la tabla siguiente se enumeran los métodos de la clase Media.

Método	Componentes	Descripción
`Media.addCuePoint()`	MediaDisplay, MediaPlayback	Añade un objeto de cuepoint a la instancia de componente.
`Media.associateController()`	MediaDisplay	Asocia una instancia de MediaDisplay con una instancia de MediaController.
`Media.associateDisplay()`	MediaController	Asocia una instancia de MediaController con una instancia de MediaDisplay.

Método	Componentes	Descripción
`Media.displayFull()`	MediaPlayback	Convierte la instancia de componente al modo de reproducción de pantalla completa.
`Media.displayNormal()`	MediaPlayback	Vuelve a convertir la instancia de componente al tamaño de pantalla original.
`Media.getCuePoint()`	MediaDisplay, MediaPlayback	Devuelve un objeto de cuepoint.
`Media.pause()`	MediaDisplay, MediaPlayback	Pone en pausa la cabeza lectora en la ubicación actual de la línea de tiempo del medio.
`Media.play()`	MediaDisplay, MediaPlayback	Reproduce el medio asociado con la instancia de componente en un punto inicial concreto.
`Media.removeAllCuePoints()`	MediaDisplay, MediaPlayback	Elimina todos los objetos de cuepoint asociados con una instancia de componente determinada.
`Media.removeCuePoint()`	MediaDisplay, MediaPlayback	Elimina un cuepoint concreto, asociado con una instancia de componente determinada.
`Media.setMedia()`	MediaDisplay, MediaPlayback	Establece el tipo de medio y la ruta del tipo de medio especificado.
`Media.stop()`	MediaDisplay, MediaPlayback	Detiene la cabeza lectora y la desplaza a la posición 0, que es el principio del medio.

Resumen de propiedades de la clase Media

En la tabla siguiente se enumeran las propiedades de la clase Media.

Propiedad	Componentes	Descripción
`Media.activePlayControl`	MediaController	Determina el estado del componente cuando éste se carga en tiempo de ejecución.
`Media.aspectRatio`	MediaDisplay, MediaPlayback	Determina si la instancia de componente mantiene su proporción de vídeo.

Propiedad	Componentes	Descripción
Media.autoPlay	MediaDisplay, MediaPlayback	Determina si la instancia de componente comenzará inmediatamente a almacenarse en el búfer y a reproducirse.
Media.autoSize	MediaDisplay, MediaPlayback	Determina el tamaño de la parte de visualización del medio del componente MediaDisplay o MediaPlayback.
Media.backgroundStyle	MediaController	Determina si la instancia de componente dibuja su fondo cromático.
Media.bytesLoaded	MediaDisplay, MediaPlayback	Sólo lectura; número de bytes cargados disponibles para reproducir.
Media.bytesTotal	MediaDisplay, MediaPlayback	Número de bytes que se van a cargar en la instancia de componente.
Media.contentPath	MediaDisplay, MediaPlayback	Cadena que contiene la ruta relativa y el nombre de archivo del medio que va a reproducirse con o sin interrupción.
Media.controllerPolicy	MediaController, MediaPlayback	Determina si se oculta el controlador cuando se crean instancias de él y sólo aparece cuando el usuario desplaza el puntero del ratón sobre el controlador en estado contraído.
Media.controlPlacement	MediaPlayback	Determina la posición de los controles del componente.
Media.cuePoints	MediaDisplay, MediaPlayback	Matriz de objetos de cuepoint que se han asignado a una instancia de componente determinada.
Media.horizontal	MediaController	Determina la orientación de la instancia de componente.
Media.mediaType	MediaDisplay, MediaPlayback	Determina el tipo de medio que va a reproducirse.
Media.playheadTime	MediaDisplay, MediaPlayback	Contiene la posición actual de la cabeza lectora (en segundos) en la línea de tiempo del medio que se está reproduciendo.

Propiedad	Componentes	Descripción
Media.playing	MediaDisplay, MediaPlayback, MediaController	Para MediaDisplay y MediaPlayback, esta propiedad es de sólo lectura y devuelve un valor booleano que indica si una determinada instancia de componente está reproduciendo un medio. Para MediaController, esta propiedad es de lectura/escritura y controla la imagen (en reproducción o pausa) que se muestra en el botón Reproducir/Pausar del controlador.
Media.preferredHeight	MediaDisplay, MediaPlayback	Valor predeterminado de la altura de un archivo FLV.
Media.preferredWidth	MediaDisplay, MediaPlayback	Valor predeterminado de la anchura de un archivo FLV.
Media.totalTime	MediaDisplay, MediaPlayback	Entero que indica la duración total en segundos del medio.
Media.volume	MediaDisplay, MediaPlayback	Entero de 0 (mínimo) a 100 (máximo) que representa el nivel de volumen.

Resumen de eventos de la clase Media

En la tabla siguiente se enumeran los eventos de la clase Media.

Evento	Componentes	Descripción
Media.change	MediaDisplay, MediaPlayback	Se difunde continuamente mientras se reproduce el medio.
Media.click	MediaController, MediaPlayback	Se difunde cuando el usuario hace clic en el botón Reproducir/Pausar.
Media.complete	MediaDisplay, MediaPlayback	Notificación de que la cabeza lectora ha alcanzado el final del medio.
Media.cuePoint	MediaDisplay, MediaPlayback	Notificación de que la cabeza lectora ha alcanzado un cuepoint determinado.
Media.playheadChange	MediaController, MediaPlayback	Difundido por la instancia de componente cuando un usuario mueve el control deslizante del reproductor o hace clic en el botón Ir al principio o Ir al final.
Media.progress	MediaDisplay, MediaPlayback	Se genera continuamente hasta que el medio está totalmente descargado.

Evento	Componentes	Descripción
`Media.scrubbing`	MediaController, MediaPlayback	Se genera cuando se arrastra la cabeza lectora.
`Media.volume`	MediaController, MediaPlayback	Se difunde cuando el usuario ajusta el volumen.

Media.activePlayControl

Se aplica a

MediaController.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.activePlayControl

Descripción

Propiedad; valor de cadena que especifica el estado en que debe encontrarse el componente MediaController al cargarse en tiempo de ejecución. Un valor `"play"` indica un estado de reproducción; un valor `"pause"` indica un estado de pausa. Establezca esta propiedad y la propiedad `autoPlay` de forma que ambas indiquen el mismo estado. El valor predeterminado es `"play"`.

La imagen del botón que se muestra en el componente MediaController es lo contrario del estado de reproducción/pausa actual. Por ejemplo, en el estado de reproducción, MediaController muestra un botón de pausa, ya que eso es lo que sucedería si el usuario hiciera clic en el botón y cambiara el estado.

Puesto que indica el estado en que se encontrará el controlador cuando se cargue, la propiedad `activePlayControl` debe establecerse antes de que se cree el controlador, ya sea mediante el inspector de propiedades o el inspector de componentes, si el componente está en el escenario. Si el componente se crea con código ActionScript, esta propiedad debe establecerse en el parámetro `initObj`. Cambiar el valor de esta propiedad tras la creación del componente no tiene ninguna consecuencia. El valor sólo puede modificarlo el usuario haciendo clic en el botón Reproducir/Pausar.

Ejemplo

En el ejemplo siguiente se indica que el control está en pausa cuando se carga por primera vez:

```
myMedia.activePlayControl = "pause";
```

Véase también

`Media.autoPlay`

Media.addCuePoint()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.addCuePoint(*cuePointName*, *cuePointTime*)

Parámetros

cuePointName Cadena que asigna un nombre al cuepoint.

cuePointTime Número que indica, en segundos, cuándo se difunde un evento `cuePoint`.

Valor devuelto

Ninguno.

Descripción

Método; añade un objeto de cuepoint a una instancia de MediaPlayback o MediaDisplay. Cuando el tiempo de la cabeza lectora y el del cuepoint son iguales, se difunde un evento `cuePoint`.

Ejemplo

En el código siguiente se añade un cuepoint llamado `Homerun` a `myMedia` cuando el tiempo de la cabeza lectora equivale a 16 segundos.

```
myMedia.addCuePoint("Homerun", 16);
```

Véase también

`Media.cuePoint`, `Media.cuePoints`, `Media.getCuePoint()`, `Media.removeAllCuePoints()`, `Media.removeCuePoint()`

Media.aspectRatio

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.aspectRatio`

Descripción

Propiedad; valor booleano que determina si una instancia de MediaDisplay o MediaPlayback mantiene su proporción de vídeo durante la reproducción. `true` indica que la proporción debe mantenerse; `false` indica que la proporción puede cambiar durante la reproducción. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se indica que la proporción puede cambiar durante la reproducción:

```
myMedia.aspectRatio = false;
```

Media.associateController()

Se aplica a

MediaDisplay.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.associateController(*instanceName*)

Parámetros

instanceName Cadena que especifica el nombre de instancia del componente MediaController que se va a asociar.

Valor devuelto

Ninguno.

Descripción

Método; asocia una instancia de MediaDisplay con una instancia de MediaController.

Si utiliza `Media.associateDisplay()` para asociar una instancia de MediaController con una instancia de MediaDisplay, no es necesario utilizar `Media.associateController()`.

Ejemplo

En el código siguiente se asocia `myMedia` con `myController`:

```
myMedia.associateController(myController);
```

Véase también

`Media.associateDisplay()`

Media.associateDisplay()

Se aplica a

MediaController.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.associateDisplay(*instanceName*)

Parámetros

instanceName Cadena que especifica el nombre de instancia del componente MediaDisplay que se va a asociar.

Valor devuelto

Ninguno.

Descripción

Método; asocia una instancia de MediaController con una instancia de MediaDisplay.

Si asocia una instancia de MediaDisplay con una instancia de MediaController mediante `Media.associateController()`, no es necesario utilizar `Media.associateDisplay()`.

Ejemplo

En el código siguiente se asocia `myMedia` con `myDisplay`:

```
myMedia.associateDisplay(myDisplay);
```

Véase también

`Media.associateController()`

Media.autoPlay

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.autoPlay`

Descripción

Propiedad; valor booleano que determina si la instancia de MediaPlayback o MediaDisplay debe comenzar inmediatamente a intentar almacenarse en el búfer y reproducirse. Un valor `true` indica que el control se almacena en el búfer y se reproduce en tiempo de ejecución; un valor `false` indica que el control se detiene en tiempo de ejecución. Esta propiedad depende de las propiedades `contentPath` y `mediaType`. Si `contentPath` y `mediaType` no se establecen, no hay reproducción en tiempo de ejecución. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se indica que el control no se inicia cuando se carga por primera vez:

```
myMedia.autoPlay = false;
```

Véase también

`Media.contentPath`, `Media.mediaType`

Media.autoSize

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.autoSize`

Descripción

Propiedad; valor booleano que determina el tamaño de la parte de visualización del medio del componente MediaDisplay o MediaPlayback.

En el componente MediaDisplay, la propiedad se comporta de la manera siguiente:

- Si establece el valor `true` para esta propiedad, Flash muestra el medio con su tamaño preferido, independientemente del tamaño del componente. Esto implica que, a menos que el tamaño de la instancia de MediaDisplay sea igual al tamaño del medio, éste rebasará los límites de la instancia o no llenará el tamaño de la misma.
- Si establece el valor `false`, Flash utiliza el tamaño de la instancia mientras sea posible, al tiempo que respeta la proporción. Si `Media.autoSize` y `Media.aspectRatio` se establecen en `false`, se utiliza el tamaño exacto del componente.

En el componente MediaPlayback, la propiedad se comporta de la manera siguiente:

- Si establece el valor `true` para esta propiedad, Flash muestra el medio con su tamaño preferido, siempre que el área de reproducción del medio sea inferior al tamaño preferido. En tal caso, Flash reduce el tamaño del medio para adaptarlo al interior de la instancia, respetando la proporción. Si el tamaño preferido es inferior al tamaño del área del medio de la instancia, parte de dicha área no se utiliza.

- Si establece el valor `false`, Flash utiliza el tamaño de la instancia mientras sea posible, al tiempo que respeta la proporción. Si `Media.autoSize` y `Media.aspectRatio` se establecen en `false`, se llena el área del medio del componente. Esta área se define como el área situada sobre los controles (en el diseño predeterminado), menos un margen de 8 píxeles circundante que forma los bordes del componente.

El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se indica que el control no vuelve a reproducirse en función del tamaño del medio:

```
myMedia.autoSize = false;
```

Véase también

`Media.aspectRatio`

Media.backgroundStyle

Se aplica a

MediaController.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

`myMedia``.backgroundStyle`

Descripción

Propiedad; valor de cadena que indica el fondo que se dibuja para la instancia de MediaController. Un valor `"default"` indica que se dibuja el fondo cromático, mientras que un valor `"none"` indica que no se dibuja dicho fondo. El valor predeterminado es `"default"`.

No se trata de una propiedad de estilo y, por lo tanto, no se ve afectada por la configuración de estilos.

Ejemplo

En el ejemplo siguiente se indica que no se dibujará el fondo cromático para el control:

```
myMedia.backgroundStyle = "none";
```

Media.bytesLoaded

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.bytesLoaded

Descripción

Propiedad de sólo lectura; el número de bytes ya cargados en el componente que están disponibles para su reproducción. El valor predeterminado es `undefined`.

Ejemplo

En el código siguiente se crea una variable llamada `PlaybackLoad` que se establecerá con el número de bytes cargados. La variable se utilizará en un bucle `for`.

```
// Crear variable que indique el número de bytes cargados.
var PlaybackLoad:Number = myMedia.bytesLoaded;
// Realizar alguna función hasta que la reproducción esté preparada.
for (PlaybackLoad < 150) {
  someFunction();
}
```

Media.bytesTotal

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.bytesTotal

Descripción

Propiedad de sólo lectura; número de bytes que se van a cargar en el componente MediaPlayback o MediaDisplay. El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se indica al usuario el tamaño del medio que se va a reproducir sin interrupción:

```
myTextField.text = myMedia.bytesTotal;
```

Media.change

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
  // Introducir aquí el código propio.
}
myMedia.addEventListener("change", listenerObject)
```

Descripción

Evento; difundido por los componentes MediaDisplay y MediaPlayback mientras se reproduce el medio. El porcentaje completo puede recuperarse de la instancia de componente.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Media.change` tiene dos propiedades adicionales:

`target` Referencia al objeto difusor.

`type` Cadena `"change"`, que indica el tipo de evento.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar la posición de la cabeza lectora (`Media.playheadTime`), a partir de la cual puede calcularse el porcentaje completo:

```
var myPlayerListener:Object = new Object();
myPlayerListener.change = function(eventObj:Object) {
  var myPosition:Number = myPlayer.playheadTime;
  var myPercentPosition:Number = (myPosition/myPlayer.totalTime);
};
myPlayer.addEventListener("change", myPlayerListener);
```

Véase también

`Media.playing`, `Media.pause()`

Media.click

Se aplica a

MediaController, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("click", listenerObject);
```

Descripción

Evento; se difunde cuando el usuario hace clic en el botón Reproducir/Pausar. El campo de detalle puede utilizarse para determinar en qué botón se ha hecho clic. El objeto del evento `Media.click` tiene las siguientes propiedades:

`detail` Cadena `"pause"` o `"play"`.

`target` Referencia a la instancia de MediaController o MediaPlayback.

`type` Cadena `"click"`.

Ejemplo

Para una instancia de componente MediaController denominada **myMedia** (con un componente Window en la biblioteca), el ejemplo siguiente abre una ventana emergente cuando el usuario hace clic en el botón Reproducir/Pausar:

```
var myMediaListener:Object = new Object();
myMediaListener.click = function(eventObj:Object) {
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
};
myMedia.addEventListener("click", myMediaListener);
```

Media.complete

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("complete", listenerObject);
```

Descripción

Evento; notificación de que la cabeza lectora ha llegado al final del medio. El objeto del evento `Media.complete` tiene las siguientes propiedades:

`target` Referencia a la instancia de MediaDisplay o MediaPlayback.

`type` Cadena `"complete"`.

Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar cuándo acaba la reproducción del medio:

```
var myListener:Object = new Object();
myListener.complete = function(eventObj:Object) {
    trace("media is Finished");
};
myMedia.addEventListener("complete", myListener);
```

Media.contentPath

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.contentPath
```

Descripción

Propiedad; cadena que contiene la ruta relativa y el nombre de archivo del medio que va a reproducirse con o sin interrupción. Establecer la propiedad `contentPath` equivale a llamar al método `Media.setMedia()` sin especificar un parámetro *`mediaType`*. Si no se establece ningún parámetro *`mediaType`* con `Media.setMedia()`, el tipo predeterminado es FLV. El valor predeterminado de la propiedad `contentPath` es `undefined`.

Ejemplo

En el ejemplo siguiente se muestra el nombre del medio que se reproduce en un cuadro de texto:

```
myTextField.text = myMedia.contentPath;
```

Véase también

`Media.setMedia()`

Media.controllerPolicy

Se aplica a

MediaController, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.controllerPolicy

Descripción

Propiedad; determina si el componente MediaController (o el subcomponente de controlador del componente MediaPlayback) se oculta cuando se crean instancias de él y sólo aparece cuando el usuario desplaza el puntero del ratón sobre el controlador en estado contraído.

Los valores posibles para esta propiedad son los siguientes:

- "on" especifica que los controles están siempre expandidos.
- "off" especifica que los controles están siempre contraídos.
- "auto" (valor predeterminado) especifica que el control permanece en estado contraído hasta que el usuario desplace el puntero del ratón sobre el área activa. El área activa coincide con el área en la que se dibuja el control contraído. El control permanece expandido hasta que el ratón sale del área activa.

NOTA	El área activa se expande y se contrae con el controlador.

Ejemplo

En el ejemplo siguiente se mantiene el controlador abierto constantemente:

```
myMedia.controllerPolicy = "on";
```

Media.controlPlacement

Se aplica a

MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.controlPlacement

Descripción

Propiedad; determina la posición de la parte de controlador del componente MediaPlayback con relación a su visualización. Los valores posibles son `"top"`, `"bottom"`, `"left"` y `"right"`. El valor predeterminado es `"bottom"`.

Ejemplo

En el ejemplo siguiente, la parte de controlador del componte MediaPlayback se sitúa en el lado derecho:

```
myMedia.controlPlacement = "right";
```

Media.cuePoint

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("cuePoint", listenerObject);
```

Descripción

Evento; notificación de que la cabeza lectora ha alcanzado el cuepoint. El objeto del evento `Media.cuePoint` tiene las siguientes propiedades:

`cuePointName` Cadena que indica el nombre del cuepoint.

`cuePointTime` Número que indica, en fotogramas o segundos, cuándo se ha alcanzado el cuepoint.

`target` Referencia al objeto MediaPlayback, si lo hay, o bien al propio objeto MediaDisplay.

`type` Cadena `"cuePoint"`.

Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar cuándo se ha alcanzado un cuepoint:

```
var myCuePointListener:Object = new Object();
myCuePointListener.cuePoint = function(eventObject:Object){
  trace("heard " + eventObject.type + ", " + eventObject.target + ", " +
  eventObject.cuePointName + ", " + eventObject.cuePointTime);
};
myPlayback.addEventListener("cuePoint", myCuePointListener);
```

Véase también

`Media.addCuePoint()`, `Media.cuePoints`, `Media.getCuePoint()`

Media.cuePoints

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.cuePoints

o

myMedia.cuePoints[*N*]

Descripción

Propiedad; matriz de objetos de cuepoint asignados a una instancia de MediaPlayback o MediaDisplay. En la matriz, cada objeto de cuepoint puede tener un nombre, un tiempo en segundos o fotogramas y una propiedad de reproductor (que es el nombre de instancia del componente con el que está asociado). El valor predeterminado es una matriz vacía (`[]`).

Ejemplo

En el ejemplo siguiente se elimina el tercer cuepoint si se reproduce la vista previa de una acción:

```
if (myVariable == actionPreview) {
  myMedia.removeCuePoint(myMedia.cuePoints[2]);
}
```

Véase también

`Media.addCuePoint()`, `Media.getCuePoint()`, `Media.removeCuePoint()`

Media.displayFull()

Se aplica a

MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.displayFull()`

Valor devuelto

Ninguno.

Descripción

Método; establece el modo de pantalla completa para la instancia de MediaPlayback. En este modo, el componente se expande para llenar todo el escenario. Para que el componente recupere su tamaño normal, utilice `Media.displayNormal()`.

Ejemplo

En el código siguiente se fuerza la expansión del componente para que ocupe el escenario:

```
myMedia.displayFull();
```

Véase también

`Media.displayNormal()`

Media.displayNormal()

Se aplica a

MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.displayNormal()
```

Valor devuelto

Ninguno.

Descripción

Método; restablece el tamaño normal de la instancia de MediaPlayback tras haberse utilizado el método `Media.displayFull()`.

Ejemplo

En el código siguiente se restablece el tamaño original de un componente MediaPlayback:

```
myMedia.displayNormal();
```

Véase también

`Media.displayFull()`

Media.getCuePoint()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.getCuePoint(*cuePointName*)

Parámetros

cuePointName Cadena proporcionada cuando se utilizó Media.addCuePoint().

Valor devuelto

Un objeto de cuepoint.

Descripción

Método; devuelve un objeto de cuepoint según su nombre de cuepoint.

Ejemplo

En el código siguiente se recupera un cuepoint llamado myCuePointName.

```
myMedia.removeCuePoint(myMedia.getCuePoint("myCuePointName"));
```

Véase también

Media.addCuePoint(), Media.cuePoint, Media.cuePoints, Media.removeCuePoint()

Media.horizontal

Se aplica a

MediaController.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.horizontal

Descripción

Propiedad; determina si el componente MediaController se muestra en posición vertical u horizontal. true indica que el componente se muestra en posición horizontal; false indica la posición vertical. Si se establece el valor false, la barra de reproducción y el deslizador del reproductor se desplazan de abajo arriba. El valor predeterminado es true.

Ejemplo

En el ejemplo siguiente se muestra el componente MediaController en posición vertical:

```
myMedia.horizontal = false;
```

Media.mediaType

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.mediaType

Descripción

Propiedad; indica el tipo de medio (FLV o MP3) que se va a reproducir. El valor predeterminado es `"FLV"`. Véase "Trabajo con vídeo" en *Utilización de Flash*.

Ejemplo

En el ejemplo siguiente se determina el tipo de medio que se reproduce actualmente:

```
var currentMedia:String = myMedia.mediaType;
```

Véase también

`Media.setMedia()`

Media.pause()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.pause()

Valor devuelto

Ninguno.

Descripción

Método; pone en pausa la cabeza lectora en la ubicación actual.

Ejemplo

En el código siguiente se pone en pausa la reproducción.

```
myMedia.pause();
```

Media.play()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.play(*startingPoint*)

Parámetros

startingPoint Entero no negativo que indica el punto inicial (en segundos) en el que el medio debe empezar a reproducirse.

Valor devuelto

Ninguno.

Descripción

Método; reproduce el medio asociado con la instancia de componente en el punto inicial especificado. El valor predeterminado es el valor actual de `playheadTime`.

Ejemplo

En el código siguiente se indica que el componente multimedia debe empezar a reproducirse en el segundo 120:

```
myMedia.play(120);
```

Véase también

`Media.pause()`

Media.playheadChange

Se aplica a

MediaController, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.playheadChange = function(eventObject){
  // Introducir aquí el código propio.
}
myMedia.addEventListener("playheadChange", listenerObject)
```

Descripción

Evento; difundido por el componente MediaController o MediaPlayback cuando el usuario mueve el deslizador del reproductor o hace clic en el botón Ir al principio o Ir al final. El objeto del evento `Media.playheadChange` tiene las siguientes propiedades:

`detail` Número que indica qué porcentaje del medio se ha reproducido.

`type` Cadena `"playheadChange"`.

Ejemplo

En el ejemplo siguiente se envía el porcentaje reproducido al panel Salida cuando el usuario deja de arrastrar la cabeza lectora:

```
var controlListen:Object = new Object();
controlListen.playheadChange = function(eventObj:Object) {
  trace(eventObject.detail);
};
myMedia.addEventListener("playheadChange", controlListen);
```

Media.playheadTime

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.playheadTime
```

Descripción

Propiedad; contiene la posición actual de la cabeza lectora (en segundos) en la línea de tiempo del medio que se está reproduciendo. El valor predeterminado es la ubicación de la cabeza lectora.

Ejemplo

En el ejemplo siguiente se establece una variable con la ubicación de la cabeza lectora, expresada en segundos:

```
var myPlayhead:Number = myMedia.playheadTime;
```

Media.playing

Se aplica a

MediaDisplay, MediaPlayback, MediaController.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.playing
```

Descripción

Propiedad; devuelve un valor booleano que indica si el medio se está reproduciendo (`true`) o está en pausa (`false`). Esta propiedad es de sólo lectura para los componentes MediaDisplay y MediaPlayback, así como de lectura/escritura para el componente MediaController.

Ejemplo

En el código siguiente se determina si el medio se está reproduciendo o si está en pausa:

```
if(myMedia.playing == true){
  some function;
}
```

Véase también

`Media.change`

Media.preferredHeight

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.preferredHeight`

Descripción

Propiedad; se establece en función del valor de altura predeterminado de un archivo FLV. Esta propiedad se aplica sólo a los medios FLV, dado que la altura de los archivos MP3 es fija. Esta propiedad puede utilizarse para establecer las propiedades de altura y anchura (más un margen para el propio componente). El valor predeterminado es `undefined` si no se define un elemento multimedia FLV.

Ejemplo

En el ejemplo siguiente se establece el tamaño de una instancia de MediaPlayback en función del medio que se está reproduciendo y cuenta con el margen de píxeles necesario para la instancia de componente:

```
if (myPlayback.contentPath != undefined) {
  var mediaHeight:Number = myPlayback.preferredHeight;
  var mediaWidth:Number = myPlayback.preferredWidth;
  myPlayback.setSize((mediaWidth + 20), (mediaHeight + 70));
}
```

Media.preferredWidth

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.preferredWidth

Descripción

Propiedad; se establece en función del valor de anchura predeterminado de un archivo FLV. El valor predeterminado es `undefined`.

Ejemplo

En el ejemplo siguiente se establece la anchura deseada de la variable `mediaWidth`:

```
var mediaWidth:Number = myMedia.preferredWidth;
```

Media.progress

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("progress", listenerObject);
```

Descripción

Evento; se genera continuamente hasta que el medio se ha descargado completamente. El objeto del evento `Media.progress` tiene las siguientes propiedades:

`target` Referencia a la instancia de MediaDisplay o MediaPlayback.

`type` Cadena `"progress"`.

Ejemplo

En el ejemplo siguiente se detecta el progreso:

```
var myProgressListener:Object = new Object();
myProgressListener.progress = function(eventObj:Object) {
  // lightMovieClip debe parpadear durante el progreso.
  var lightVisible:Boolean = lightMovieClip.visible;
  lightMovieClip.visible = !lightVisible;
};
```

En el siguiente ejemplo se detecta el progreso y se llama a otra función si el evento progress se prolonga durante más de 3000 milisegundos (3 segundos):

```
// Duración de demora antes de llamar a timeOut.
var timeOut:Number = 3000;

// Si se ha alcanzado timeOut, realizar la siguiente operación:
function callback(arg) {
  trace(arg);
}

// Detectar progreso.
var myListener:Object = new Object();
myListener.progress = function(eventObj:Object) {
  setInterval(callback, timeOut, "Experiencing Network Delay");
};
md.addEventListener("progress", myListener);
```

Media.scrubbing

Se aplica a

MediaController, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("scrubbing", listenerObject);
```

Descripción

Evento; se genera cuando se arrastra la cabeza lectora.

`target` Referencia a la instancia de MediaController o MediaPlayback.

`type` Cadena `"scrubbing"`.

Ejemplo

En el siguiente ejemplo se detecta el usuario que arrastra la cabeza lectora:

```
my_mp.addEventListener("scrubbing", scrubbingListener);
function scrubbingListener(evt_obj:Object):Void {
  trace(evt_obj.type+" @ "+getTimer()+" ms
  (isScrubbing="+evt_obj.detail+")");
}
```

Media.removeAllCuePoints()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.removeAllCuePoints()

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los objetos de cuepoint asociados con una instancia de componente.

Ejemplo

En el código siguiente se eliminan todos los objetos de cuepoint:

```
myMedia.removeAllCuePoints();
```

Véase también

Media.addCuePoint(), Media.cuePoints, Media.removeCuePoint()

Media.removeCuePoint()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.removeCuePoint(*cuePoint*)

Parámetros

cuePoint Referencia a un objeto de cuepoint asignado previamente mediante Media.addCuePoint().

Valor devuelto

Ninguno.

Descripción

Método; elimina un cuepoint asociado con una instancia de componente.

Ejemplo

En el código siguiente se elimina un cuepoint llamado `myCuePoint`:

```
myMedia.removeCuePoint(getCuePoint("myCuePoint"));
```

Véase también

`Media.addCuePoint()`, `Media.cuePoints`, `Media.removeAllCuePoints()`

Media.setMedia()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.setMedia(contentPath [, mediaType])
```

Parámetros

`contentPath` Cadena que indica la URL del medio que se va a reproducir. El valor predeterminado es `undefined`.

`mediaType` Cadena utilizada para definir el tipo de elemento multimedia como FLV o MP3. Este parámetro es opcional. El valor predeterminado es FLV.

Valor devuelto

Ninguno.

Descripción

Método; define mediante un parámetro URL el tipo de medio y la ruta del tipo de medio especificado.

Este método es el procedimiento recomendado para establecer la ruta del contenido y el tipo de medio para los componentes MediaPlayback y MediaDisplay. La propiedad `Media.contentPath` también se puede utilizar para establecer la ruta del contenido, pero no permite definir el tipo de medio.

Si sólo trabaja con archivos FLV, no es necesario que especifique un parámetro *mediaType*. Si trabaja exclusivamente con archivos MP3, debe establecer el parámetro *mediaType* en MP3 una vez. Si alterna archivos FLV y MP3, debe cambiar el tipo de medio cada vez en la llamada `setMedia()`. Si intenta reproducir un archivo MP3 sin establecer explícitamente el tipo de medio en MP3, no se reproducirá el archivo.

Ejemplo

En el código siguiente se proporciona un medio nuevo para reproducir una instancia de componente:

```
myMedia.setMedia("http://www.helpexamples.com/flash/video/clouds.flv",
  "FLV");
```

Media.stop()

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia`.stop()`

Valor devuelto

Ninguno.

Descripción

Método; detiene la cabeza lectora y la desplaza a la posición 0, que es el principio del medio.

Ejemplo

En el código siguiente se detiene la cabeza lectora y se desplaza a la posición 0:

```
myMedia.stop()
```

Media.totalTime

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
myMedia.totalTime
```

Descripción

Propiedad; duración total del medio, expresada en segundos. Dado que el formato de archivo FLV no proporciona su tiempo de reproducción a un componente multimedia hasta que se ha cargado completamente, debe introducir manualmente `Media.totalTime` para que la barra de reproducción pueda reflejar de forma precisa el tiempo de reproducción real del medio. El valor predeterminado de los archivos MP3 es el tiempo de reproducción del medio. En los archivos FLV, el valor predeterminado es `undefined`.

Esta propiedad no puede establecerse para los archivos MP3, ya que la información se encuentra en el objeto Sound.

Ejemplo

En el ejemplo siguiente se establece el tiempo de reproducción (expresado en segundos) del medio FLV:

```
myMedia.totalTime = 151;
```

Media.volume

Se aplica a

MediaDisplay, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

myMedia.volume

Descripción

Propiedad; almacena un entero que indica la configuración del volumen, que puede oscilar entre 0 y 100. El valor predeterminado de esta propiedad es 75.

Ejemplo

En el ejemplo siguiente se establece el volumen máximo de reproducción del medio:

```
myMedia.volume = 100;
```

Véase también

Media.volume, Media.pause()

Media.volume

Se aplica a

MediaController, MediaPlayback.

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.volume = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("volume", listenerObject);
```

Descripción

Evento; se difunde cuando el usuario ajusta el valor del volumen. El objeto del evento Media.volume tiene las siguientes propiedades:

`detail` Entero entre 0 y 100 que representa el nivel del volumen.

`type` Cadena `"volume"`.

Ejemplo

En el ejemplo siguiente se notifica al usuario que se ajusta el volumen:

```
var myVolListener:Object = new Object();
myVolListener.volume = function(eventObj:Object) {
   mytextfield.text = "Volume adjusted!";
};
myMedia.addEventListener("volume", myVolListener);
```

Véase también

Media.volume

CAPÍTULO 30

Componente Menu (sólo en Flash Professional)

30

El componente Menu permite a los usuarios seleccionar un elemento de un menú emergente, como por ejemplo, el menú Archivo o Edición de la mayoría de las aplicaciones de software.

Normalmente, un componente Menu se abre en una aplicación cuando el usuario desplaza el puntero sobre un activador de menús con forma de botón o hace clic en éste. También puede configurar en un script que el componente Menu se abra cuando un usuario presione una tecla determinada.

Los componentes Menu se crean siempre de forma dinámica en tiempo de ejecución. Debe arrastrar el componente desde el panel Componentes a la biblioteca; a continuación, utilice el siguiente código para crear un menú con ActionScript:

```
var myMenu = mx.controls.Menu.createMenu(parent, menuDataProvider);
```

Utilice el código siguiente para abrir un menú en una aplicación:

```
myMenu.show(x, y);
```

Un evento `menuShow` se difunde a todos los detectores de la instancia de Menu inmediatamente antes de que se muestre el menú, para poder actualizar el estado de los elementos de menú. Del mismo modo, inmediatamente después de ocultarse una instancia de Menu, se difunde un evento `menuHide`.

Los elementos de un menú se describen mediante XML. Para más información, consulte "Aspectos básicos del componente Menu: vista y datos" en la página 920.

Los lectores de pantalla no pueden acceder al componente Menu.

Los menús a menudo se anidan en barras de menú. Para más información sobre barras de menús, consulte "Componente MenuBar (sólo en Flash Professional)" en la página 981.

Interacción con el componente Menu (sólo en Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente Menu.

Tras abrir un componente Menu, éste permanece visible hasta que se cierra mediante un script o hasta que el usuario hace clic con el ratón fuera del menú o en un elemento activado.

Al hacer clic se selecciona un elemento de menú, excepto los siguientes tipos de elementos de menú:

Elementos desactivados o separadores Desplazar el puntero del ratón sobre ellos o hacer clic no tiene ningún efecto (el menú permanece visible).

Anclajes de submenú Al desplazar el puntero del ratón sobre ellos se activa el submenú; los clics no tienen ningún efecto. Al desplazar el puntero sobre cualquier elemento que no sea del submenú, éste se cierra.

Al seleccionar un elemento, se envía un evento `Menu.change` a todos los detectores del menú, el menú se oculta y tienen lugar las siguientes acciones en función del tipo de elemento:

check Se activa o desactiva el atributo `selected` del elemento.

radio El elemento pasa a ser la selección actual del grupo de opciones correspondiente.

Al mover el ratón, se activan los eventos `Menu.rollOut` y `Menu.rollOver`.

Al presionar el ratón fuera del menú, se cierra el menú y se activa un evento `Menu.menuHide`.

Al liberar el ratón en un elemento activado, los tipos de elemento se ven afectados como se indica a continuación:

check Se activa o desactiva el atributo `selected` del elemento.

radio El atributo `selected` del elemento se establece en `true`, y el atributo `selected` del elemento previamente seleccionado en el grupo de opciones se establece en `false`. La propiedad `selection` del objeto de grupo de opciones correspondiente se establece con un valor que hace referencia al elemento de menú seleccionado.

Elemento no definido y nivel superior de un menú jerárquico Se activa o desactiva la visibilidad del menú jerárquico.

Cuando se selecciona una instancia de Menu al hacer clic en ella o mediante la tabulación, puede utilizar las teclas siguientes para controlarla:

Tecla	Descripción
Flecha abajo Flecha arriba	Desplaza la selección hacia abajo y hacia arriba por las filas del menú. La selección pasa de la última fila a la primera y viceversa.
Flecha derecha	Abre un submenú o desplaza la selección al menú siguiente de una barra de menús (si existe).
Flecha izquierda	Cierra un submenú y vuelve a seleccionar el menú principal (si existe), o desplaza la selección al menú anterior de una barra de menús (si existe).
Intro	Abre un submenú. Si no existe, esta clave tiene el mismo efecto que hacer clic en una fila y soltar el botón del ratón.

NOTA

Si hay un menú abierto, puede presionar el tabulador para salir de él. Debe realizar una selección o cerrar el menú presionando la tecla Esc.

Utilización del componente Menu (sólo en Flash Professional)

Puede utilizar el componente Menu para crear menús de opciones seleccionables; es como el menú Archivo o Edición de la mayoría de las aplicaciones de software. También puede utilizar el componente Menu para crear menús contextuales que aparecen cuando un usuario presiona una zona interactiva o una tecla modificadora. Utilice el componente Menu con el componente MenuBar para crear una barra de menús horizontal con menús que se amplían bajo cada elemento de la barra de menús.

Al igual que los menús de escritorio estándar, el componente Menu admite elementos de menú cuyas funciones se dividen en las siguientes categorías generales:

Activadores de comandos Estos elementos activan eventos; estos eventos se gestionan con código.

Anclajes de submenús Estos elementos son anclajes que abren submenús.

Botones de opción Estos elementos funcionan en grupos; sólo pueden seleccionarse de uno en uno.

Elementos de casilla de verificación Estos elementos representan un valor booleano (`true` o `false`).

Separadores Estos elementos proporcionan una línea horizontal sencilla que divide los elementos de un menú en distintos grupos visuales.

Aspectos básicos del componente Menu: vista y datos

Conceptualmente, el componente Menu consta de un modelo de datos y una vista que muestra los datos. La clase Menu proporciona la vista y contiene los métodos de configuración visual. La Clase MenuDataProvider añade métodos al objeto prototipo XML global (de modo muy parecido a como lo hace la API DataProvider con el objeto Array); estos métodos permiten construir externamente proveedores de datos y añadirlos a varias instancias de menú. El proveedor de datos difunde los cambios realizados a todas las vistas de cliente. (Véase "Clase MenuDataProvider" en la página 970.)

Una instancia Menu es una colección jerárquica de elementos XML que se corresponde con elementos de menú individuales. Los atributos definen el comportamiento y el aspecto del elemento de menú correspondiente que aparece en la pantalla. La colección puede convertirse fácilmente a XML o de XML, que sirve para describir menús (la etiqueta `menu`) y elementos (la etiqueta `menuitem`). La clase XML incorporada de ActionScript es la base del modelo subyacente del componente Menu.

Un menú sencillo de dos elementos puede describirse en XML con dos subelementos de menú:

```
<menu>
  <menuitem label="Up" />
  <menuitem label="Down" />
</menu>
```

NOTA Los nombres de etiqueta de los nodos XML (`menu` y `menuitem`) no son importantes; en el menú se utilizan los atributos y sus relaciones de anidación.

Menús jerárquicos

Para crear menús jerárquicos, incorpore elementos XML en un elemento XML principal, de la siguiente manera:

```
<menu>
    <menuitem label="MenuItem A" >
      <menuitem label="SubMenuItem 1-A" />
      <menuitem label="SubMenuItem 2-A" />
    </menuitem>
    <menuitem label="MenuItem B" >
      <menuitem label="SubMenuItem 1-B" />
      <menuitem label="SubMenuItem 2-B" />
    </menuitem>
</menu>
```

Esto convierte al elemento de menú principal en un anclaje del menú emergente, que no genera eventos cuando se selecciona.

Atributos XML de elementos de menú

Los atributos de un elemento XML de menú determinan el contenido que se va visualizar, el comportamiento del elemento de menú y cómo se expone en ActionScript. En la tabla siguiente se describen los atributos de un elemento de menú XML:

Nombre de atributo	Tipo	Valor predeterminado	Descripción
`label`	String	`undefined`	Texto que se visualiza para representar un elemento de menú. Este atributo es necesario para todos los tipos de elementos, excepto `separator`.
`type`	`separator`, `check`, `radio`, `normal` o `undefined`	`undefined`	Tipo de elemento de menú: `separator`, `check box`, `radio button` o `normal` (un activador de comandos o submenús). Si este atributo no existe, el valor predeterminado es `normal`.
`icon`	String	`undefined`	Identificador de vinculación de un activo de imagen. Este atributo no es necesario ni está disponible para los tipos `check`, `radio` o `separator`.
`instanceName`	String	`undefined`	Identificador que puede utilizarse para hacer referencia a la instancia del elemento de menú desde la instancia del menú raíz. Por ejemplo, puede hacerse referencia a un elemento de menú llamado *yellow* como `myMenu.yellow`. Este atributo no es necesario.
`groupName`	String	`undefined`	Identificador que puede utilizarse para asociar varios elementos de botón de opción en un grupo de opciones y para exponer el estado de un grupo de opciones desde la instancia del menú raíz. Por ejemplo, puede hacerse referencia a un grupo de opciones llamado *colors* como `myMenu.colors`. Este atributo sólo es necesario para el tipo `radio`.

Nombre de atributo	Tipo	Valor predeterminado	Descripción
`selected`	Valor booleano (`false` o `true`) o cadena (`"false"` o `"true"`)	`false`	Valor booleano o cadena que indica si un elemento `check` o `radio` está activado (`true`) o desactivado (`false`). Este atributo no es necesario.
`enabled`	Valor booleano (`false` o `true`) o cadena (`"false"` o `"trueF`)	`true`	Valor booleano o cadena que indica si este elemento de menú puede seleccionarse (`true`) o no (`false`). Este atributo no es necesario.

Tipos de elementos de menú (sólo en Flash Professional)

Hay cuatro tipos de elementos de menú, especificados por el atributo `type`:

```
<menu>
    <menuitem label="Normal Item" />
    <menuitem type="separator" />
    <menuitem label="Checkbox Item" type="check" instanceName="check_1"/>
    <menuitem label="RadioButton Item" type="radio"
  groupName="radioGroup_1" />
</menu>
```

Elementos de menú normales

El elemento de menú `Normal Item` no tiene un atributo `type`, lo cual significa que el atributo `type` tiene el valor predeterminado `normal`. Los elementos normales pueden ser activadores de comandos o submenús, según si tienen subelementos anidados.

Elementos de menú separadores

El elemento de menú cuyo atributo `type` es `separator` actúa como divisor visual de un menú. En el siguiente XML se crean tres elementos de menú, Top, Middle y Bottom, con separadores entre sí:

```
<menu>
    <menuitem label="Top" />
    <menuitem type="separator" />
    <menuitem label="Middle" />
    <menuitem type="separator" />
    <menuitem label="Bottom" />
</menu>
```

Todos los elementos separadores están desactivados. Hacer clic en un separador o desplazar el puntero sobre éste no tiene efecto alguno.

Elementos de menú de casilla de verificación

El elemento de menú cuyo atributo `type` es `check` actúa como elemento de casilla de verificación en el menú. Si el atributo `selected` tiene el valor `true`, aparece una marca de verificación junto a la etiqueta del elemento de menú. Cuando un elemento de casilla de verificación se selecciona, su estado cambia automáticamente y se difunde un evento `change` a todos los detectores del menú raíz. Sin embargo, aunque un elemento de menú de casilla de verificación se comporta de forma similar a un componente CheckBox, un elemento de menú de casilla de verificación aparece visualmente sin el cuadro que rodea la verificación. Por tanto, un elemento de menú de casilla de verificación sin marcar parece un elemento de menú normal hasta que se selecciona.

En el ejemplo siguiente se definen tres elementos de menú de casillas de verificación:

```
<menu>
    <menuitem label="Apples" type="check" instanceName="buyApples"
  selected="true" />
    <menuitem label="Oranges" type="check" instanceName="buyOranges"
  selected="false" />
    <menuitem label="Bananas" type="check" instanceName="buyBananas"
  selected="false" />
</menu>
```

Se pueden utilizar los nombres de instancia de ActionScript para tener acceso a los elementos de menú directamente desde el propio menú, como se indica en el ejemplo siguiente:

```
myMenu.setMenuItemSelected(myMenu.buyapples, true);
myMenu.setMenuItemSelected(myMenu.buyoranges, false);
```

NOTA El atributo `selected` sólo debe modificarse con el método `setMenuItemSelected()`. Se puede examinar directamente el atributo `selected`, pero devuelve el valor de cadena `true` o `false`.

Elementos de menú de botón de opción

Los elementos de menú cuyo atributo `type` es `radio` pueden agruparse de modo que pueda seleccionarse sólo un elemento cada vez. Aunque un elemento de menú de botón de opción se comporta de forma similar a un componente RadioButton, un elemento de menú de botón de opción aparece visualmente sin el borde que rodea el botón. Por tanto, un elemento de menú de botón de opción sin marcar parece un elemento de menú normal hasta que se selecciona.

Un grupo de opciones se crea indicando el mismo valor en el atributo `groupName` de los elementos de menú, como en el ejemplo siguiente:

```
<menu>
    <menuitem label="Center" type="radio" groupName="alignment_group"
       instanceName="center_item"/>
    <menuitem type="separator" />
    <menuitem label="Top"    type="radio" groupName="alignment_group" />
    <menuitem label="Bottom" type="radio" groupName="alignment_group" />
    <menuitem label="Right"  type="radio" groupName="alignment_group" />
    <menuitem label="Left"   type="radio" groupName="alignment_group" />
</menu>
```

Cuando el usuario selecciona uno de los elementos, la selección actual cambia automáticamente y se difunde un evento `change` a todos los detectores del menú raíz. El elemento seleccionado actualmente de un grupo de opciones está disponible en ActionScript con la propiedad `selection`, como se indica a continuación:

```
var selectedMenuItem = myMenu.alignment_group.selection;
myMenu.alignment_group = myMenu.center_item;
```

Cada uno de los valores `groupName` debe ser exclusivo en el ámbito de la instancia del menú raíz.

NOTA El atributo `selected` sólo debe modificarse con el método `setMenuItemSelected()`. Se puede examinar directamente el atributo `selected`, pero devuelve el valor de cadena `true` o `false`.

Exposición de elementos de menú en ActionScript

Puede asignar un identificador exclusivo a cada elemento de menú con el atributo `instanceName`; de esta forma, se puede acceder directamente al elemento de menú desde el menú raíz. Por ejemplo, en el código XML siguiente se proporcionan atributos `instanceName` a cada elemento de menú:

```
<menu>
    <menuitem label="Item 1" instanceName="item_1" />
    <menuitem label="Item 2" instanceName="item_2" >
      <menuitem label="SubItem A" instanceName="sub_item_A" />
      <menuitem label="SubItem B" instanceName="sub_item_B" />
    </menuitem>
</menu>
```

Puede utilizar ActionScript para acceder directamente a las instancias correspondientes y a sus atributos desde el componente Menu, como se indica a continuación:

```
var aMenuItem = myMenu.item_1;
myMenu.setMenuItemEnabled(item_2, true);
var aLabel = myMenu.sub_item_A.attributes.label;
```

NOTA

Cada atributo `instanceName` debe ser exclusivo en el ámbito de la instancia del componente del menú raíz (incluidos todos los submenús del menú raíz).

Propiedades de objeto de inicialización (sólo en Flash Professional)

El parámetro *`initObject`* (objeto de inicialización) es un concepto fundamental en la creación del diseño del componente Menu. El parámetro es un objeto con propiedades. Cada propiedad representa uno de los posibles atributos XML de un elemento de menú. Para obtener una descripción de las propiedades permitidas en el parámetro *`initObject`*, consulte "Atributos XML de elementos de menú" en la página 921.

El parámetro *`initObject`* se utiliza en los métodos siguientes:

- `Menu.addMenuItem()`
- `Menu.addMenuItemAt()`
- `MenuDataProvider.addMenuItem()`
- `MenuDataProvider.addMenuItemAt()`

En el ejemplo siguiente se crea un parámetro *`initObject`* con dos propiedades, `label` e `instanceName`:

```
var i = myMenu.addMenuItem({label:"myMenuItem",
  instanceName:"myFirstItem"});
```

Varias propiedades se combinan para crear un tipo concreto de elemento de menú. Para crear determinados tipos de elementos de menús (normal, separador, casilla de verificación o botón de opción) es necesario asignar propiedades específicas.

Por ejemplo, puede inicializar un elemento de menú normal con el parámetro *initObject* siguiente:

```
myMenu.addMenuItem({label:"myMenuItem", enabled:true, icon:"myIcon",
  instanceName:"myFirstItem"});
```

Puede inicializar un elemento de menú separador con el parámetro *initObject* siguiente:

```
myMenu.addMenuItem({type:"separator"});
```

Puede inicializar un elemento de casilla de verificación con el parámetro *initObject* siguiente:

```
myMenu.addMenuItem({type:"check", label:"myMenuCheck", enabled:false,
  selected:true, instanceName:"myFirstCheckItem"})
```

Puede inicializar un elemento de botón de opción con el parámetro *initObject* siguiente:

```
myMenu.addMenuItem({type:"radio", label:"myMenuRadio1", enabled:true,
  selected:false, groupName:"myRadioGroup",
  instanceName:"myFirstRadioItem"})
```

Debe tratar los atributos `instanceName`, `groupName` y `type` de un elemento de menú como de sólo lectura. Sólo debe definirlos al crear un elemento (por ejemplo, en una llamada a `addMenuItem()`). Si modifica estos atributos después de crearlos, los resultados pueden ser imprevisibles.

Parámetros de Menu (sólo en Flash Professional)

También puede establecer el siguiente parámetro de edición para cada instancia del componente Menu en el inspector de propiedades:

rowHeight indica la altura de cada una de las filas, en píxeles. Al modificar el tamaño de fuente no se altera la altura de la fila. El valor predeterminado es 20.

Utilice ActionScript para controlar el componente Menu con sus propiedades, métodos y eventos. Para más información, consulte “Clase Menu (sólo en Flash Professional)” en la página 936.

Creación de una aplicación con el componente Menu (sólo en Flash Professional)

En el ejemplo siguiente, un desarrollador está creando una aplicación y utiliza el componente Menu para exponer algunos de los comandos que el usuario puede emitir como, por ejemplo, Open, Close y Save.

Para crear una aplicación con el componente Menu:

1. Seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente Menu desde el panel Componentes a la biblioteca.

   Los menús se crean de forma dinámica con ActionScript.
3. Arrastre un componente Button desde el panel Componentes a la biblioteca.

   El botón se utilizará para activar el menú.
4. En el panel Acciones, en el primer fotograma, introduzca el código siguiente para añadir un detector de eventos que detecte los eventos `click` en el botón. El código también detecta un evento `change` en el menú y muestra el nombre del elemento de menú seleccionado en el panel Salida:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
  - Componente Button en la biblioteca
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "menu_button", 10, {label:"Launch
  Menu"});

// Crear un menú.
var my_menu:Menu = Menu.createMenu();

// Añadir algunos elementos de menú.
my_menu.addMenuItem("Open");
my_menu.addMenuItem("Close");
my_menu.addMenuItem("Save");
my_menu.addMenuItem("Revert");
```

```
// Añadir un detector de cambios al componente Menu para detectar el
  elemento de menú que se selecciona.
var menuListener:Object = new Object();
menuListener.change = function(evt_obj:Object) {
 var item_obj:Object = evt_obj.menuItem;
 trace("Item selected: "+item_obj.attributes.label);
};
my_menu.addEventListener("change", menuListener);

// Añadir un detector de botón que muestre el menú cuando se haga clic en
  el botón.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 var my_button:Button = evt_obj.target;
 // Mostrar el menú en la parte inferior del botón.
 my_menu.show(my_button.x, my_button.y + my_button.height);
};
menu_button.addEventListener("click", buttonListener);
```

5. Seleccione Control > Probar película.

   Haga clic en el botón Launch Menu para que aparezca el menú. Si selecciona un elemento de menú, una sentencia `trace()` indica la selección en el panel Salida.

Para utilizar datos XML desde un servidor a fin de crear y rellenar un menú:

1. Seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente Menu desde el panel Componentes a la biblioteca.

   Los menús se crean de forma dinámica con ActionScript.
3. En el panel Acciones, añada el siguiente código al primer fotograma para crear un menú y utilice la propiedad `dataProvider` para cargar elementos de menú desde una página Web:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

var my_menu:Menu = Menu.createMenu();

// Importar un archivo XML.
var myDP_xml:XML = new XML();
myDP_xml.ignoreWhite = true;
myDP_xml.onLoad = function(success:Boolean) {
 // Cuando llegan los datos, se pasan al menú.
 if (success) {
  my_menu.dataProvider = myDP_xml.firstChild;
 }
};
myDP_xml.load("http://www.flash-mx.com/mm/xml/menu.xml");
```

```
// Mostrar y colocar menús.
my_menu.show(100, 20);
```

NOTA	Los elementos de menú se describen mediante los elementos secundarios del primer elemento secundario del documento XML.

4. Seleccione Control > Probar película.

   A continuación se proporciona la definición de menú xml de la página Web como referencia:

```
<?xml version="1.0" ?>
<menu>
<menuitem label="Undo" />
<menuitem type="separator" />
<menuitem label="Cut" />
<menuitem label="Copy" />
<menuitem label="Paste" />
<menuitem label="Clear" />
<menuitem type="separator" />
<menuitem label="Select All" />
</menu>
```

Para crear una cadena XML correcta para crear y rellenar un menú:

1. Seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente Menu desde el panel Componentes a la biblioteca.

   Los menús se crean de forma dinámica con ActionScript.
3. En el panel Acciones, añada el código siguiente al primer fotograma para crear un menú y añada algunos elementos:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var theMenuElement_obj:Object = my_xml.addMenuItem("XXXXX");
```

```
// Añadir elementos de menú.
theMenuElement_obj.addMenuItem({label:"Undo"});
theMenuElement_obj.addMenuItem({type:"separator"});
theMenuElement_obj.addMenuItem({label:"Cut"});
theMenuElement_obj.addMenuItem({label:"Copy"});
theMenuElement_obj.addMenuItem({label:"Paste"});
theMenuElement_obj.addMenuItem({label:"Clear", enabled:"false"});
theMenuElement_obj.addMenuItem({type:"separator"});
theMenuElement_obj.addMenuItem({label:"Select All"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, theMenuElement_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);
```

4. Seleccione Control > Probar película.

Para utilizar la clase MenuDataProvider a fin de crear y rellenar un menú:

1. Seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente Menu desde el panel Componentes a la biblioteca.

   Los menús se crean de forma dinámica con ActionScript.
3. En el panel Acciones, añada el código siguiente al primer fotograma para crear un menú y añada algunos elementos:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var xml = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var theMenuElement = xml.addMenuItem("XXXXX");
```

```
// Añadir elementos de menú.
theMenuElement.addMenuItem({label:"Undo"});
theMenuElement.addMenuItem({type:"separator"});
theMenuElement.addMenuItem({label:"Cut"});
theMenuElement.addMenuItem({label:"Copy"});
theMenuElement.addMenuItem({label:"Paste"});
theMenuElement.addMenuItem({label:"Clear", enabled:"false"});
theMenuElement.addMenuItem({type:"separator"});
theMenuElement.addMenuItem({label:"Select All"});
// Crear el objeto Menu.
var my_menu = mx.controls.Menu.createMenu(_root, theMenuElement);

my_menu.show(100, 20);
```

4. Seleccione Control > Probar película.

Personalización del componente Menu (sólo en Flash Professional)

El menú adapta su tamaño horizontal al del texto más extenso. Asimismo puede llamar al método `setSize()` para adaptar el tamaño del componente. El tamaño máximo permitido de los iconos es de 16 x 16 píxeles.

Utilización de estilos con el componente Menu

Llame al método `setStyle()` para cambiar el estilo de los menús, sus elementos y sus submenús. El componente Menu admite los estilos siguientes:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`alternatingRowColors`	Ambos	Especifica los colores de las filas en un patrón alterno. El valor puede ser una matriz de más de dos colores, por ejemplo, 0xFF00FF, 0xCC6699 y 0x996699. A diferencia de los estilos de color de un solo valor, `alternatingRowColors` no acepta nombres de color; los valores deben ser códigos de color numéricos. De forma predeterminada, el estilo no está definido y `backgroundColor` se usa en su lugar para todas las filas.

Estilo	Tema	Descripción
backgroundColor	Ambos	Color de fondo del menú. El color predeterminado es blanco y se define en la declaración de estilo de clase. Este estilo se omite si se especifica alternatingRowColors.
backgroundDisabledColor	Ambos	Color de fondo cuando la propiedad del componente enabled está establecida en "false". El valor predeterminado es 0xDDDDDD (gris medio).
borderStyle	Ambos	El componente Menu utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es "menuBorder".
color	Ambos	Color del texto.
disabledColor	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
embedFonts	Ambos	Valor booleano que indica si la fuente especificada en fontFamily es una fuente incorporada. Este estilo debe definirse como true si fontFamily hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como true y fontFamily no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es false.
fontFamily	Ambos	Nombre de la fuente del texto. El valor predeterminado es "_sans".
fontSize	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
fontStyle	Ambos	Estilo de la fuente: puede ser "normal" o "italic". El valor predeterminado es "normal".
fontWeight	Ambos	Grosor de la fuente: puede ser "none" o "bold". El valor predeterminado es "none". Todos los componentes pueden aceptar además el valor "normal" en lugar de "none" durante una llamada a setStyle(), pero las llamadas posteriores a getStyle() devolverán "none".
textAlign	Ambos	Alineación del texto: puede ser "left", "right" o "center". El valor predeterminado es "left".

Estilo	Tema	Descripción
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.
`defaultIcon`	Ambos	Nombre del icono predeterminado que se mostrará en cada fila. El valor predeterminado es `undefined`, el cual indica que no se muestra ningún icono. No se necesita la propiedad del icono, no funciona para los elementos "check", "radio" o "separator", y utiliza el identificador de vinculación de un activo de imagen como parámetro de valor. Todos los elementos de menú muestran el mismo icono.
`popupDuration`	Ambos	Duración de la transición durante la apertura de un menú. El valor se especifica en milisegundos; 0 indica que no hay transición. El valor predeterminado es 150.
`rollOverColor`	Ambos	Color de fondo de una fila por la que se desplaza el puntero. El valor predeterminado es 0xE3FFD6 (verde brillante) con el tema Halo y 0xAAAAAA (gris claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a setStyle(), el marco establece `rollOverColor` en un valor relacionado con el valor de `themeColor` elegido.
`selectionColor`	Ambos	Color de fondo de una fila seleccionada. El valor predeterminado es 0xCDFFC1 (verde claro) con el tema Halo y 0xEEEEEE (gris muy claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a setStyle(), el marco establece `selectionColor` en un valor relacionado con el valor de `themeColor` elegido.
`selectionDuration`	Ambos	Duración de la transición del estado normal al estado seleccionado, expresado en milisegundos. El valor predeterminado es 200.

Estilo	Tema	Descripción
`selectionEasing`	Ambos	Referencia a la ecuación de suavizado que se utiliza para controlar la transición entre los estados de selección. La ecuación predeterminada usa una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.
`textRollOverColor`	Ambos	Color del texto cuando el puntero se desplaza sobre él. El valor predeterminado es 0x2B333C (gris oscuro). Este estilo es importante cuando se establece `rollOverColor` porque ambos colores deben complementarse para que el texto pueda verse con claridad al desplazar el puntero sobre él.
`textSelectedColor`	Ambos	Color del texto de la fila seleccionada. El valor predeterminado es 0x005F33 (gris oscuro). Este estilo es importante cuando se establece `selectionColor` porque ambos colores deben complementarse para que el texto pueda verse con claridad cuando se selecciona.
`useRollOver`	Ambos	Determina si el resaltado se activa cuando el puntero se desplaza sobre una fila. El valor predeterminado es true.

Definición de estilos de todos los componentes Menu de un documento

La clase Menu hereda de la clase ScrollSelectList. Las propiedades de estilo de clase predeterminadas se definen en la clase ScrollSelectList, que comparten todos los componentes basados en List. Puede definir directamente en esta clase nuevos valores de estilo predeterminados y los nuevos valores se reflejan en todos los componentes afectados.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Para definir una propiedad de estilo sólo en el componente Menu, puede crear una nueva instancia `CSSStyleDeclaration` y almacenarla en `_global.styles.Menu`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.Menu == undefined) {
  _global.styles.Menu = new CSSStyleDeclaration();
}
_global.styles.Menu.setStyle("backgroundColor", 0xFF00AA);
```

Cuando se crea una nueva declaración de estilo de clase, se pierden todos los valores predeterminados proporcionados por la declaración `ScrollSelectList`. Entre ellos, el valor de `backgroundColor`, necesario para la compatibilidad con eventos de ratón. Para crear una declaración de estilo de clase y conservar los valores predeterminados, utilice un bucle `for..in` para copiar los valores anteriores a la nueva declaración.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.Menu;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Para más información sobre los estilos de clase, consulte "Definición de los estilos de una clase de componente" en *Utilización de componentes*.

Utilización de aspectos con el componente Menu

El componente Menu utiliza una instancia de RectBorder para definir el borde (véase "Clase RectBorder" en la página 1103).

El componente Menu tiene activos visuales para los gráficos de ramas, marcas de verificación, puntos de botones de opción y separadores. Los aspectos no pueden aplicarse dinámicamente a dichos activos, pero éstos pueden copiarse de la carpeta Flash UI Components 2/Themes/MMDefault/Menu Assets/States en los dos temas y modificarse de la forma deseada. Los identificadores de vinculación no pueden modificarse y todas las instancias de Menu deben utilizar los mismos símbolos.

Para crear símbolos de clip de película para activos de Menu:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta Menu Assets a la biblioteca del documento.
4. Expanda la carpeta Menu Assets/States en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo MenuCheckEnabled.
6. Personalice el símbolo como desee.

   Por ejemplo, cambie la imagen de la marca de verificación por una X.

7. Repita los pasos 6 y 7 en todos los símbolos que desee personalizar.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente Menu desde el panel Componentes a la biblioteca del documento actual.

   De este modo, el componente se añade a la biblioteca y está disponible en tiempo de ejecución.
10. Añada código ActionScript a la línea de tiempo principal para crear una instancia de Menu en tiempo de ejecución:

```
var myMenu = mx.controls.Menu.createMenu();
myMenu.addMenuItem({label: "One", type: "check", selected: true});
myMenu.addMenuItem({label: "Two", type: "check"});
myMenu.addMenuItem({label: "Three", type: "check"});
myMenu.show(0, 0);
```

11. Seleccione Control > Probar película.

Clase Menu (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > ScrollSelectList > Menu

Nombre de clase de ActionScript mx.controls.Menu

Los métodos y las propiedades de la clase Menu permiten crear y editar menús en tiempo de ejecución.

Si una propiedad de la clase Menu se define con ActionScript, se sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.Menu.version);
```

NOTA	El código `trace(myMenuInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase Menu

En la tabla siguiente se enumeran los métodos de la clase Menu.

Método	Descripción
`Menu.addMenuItem()`	Añade un elemento al menú.
`Menu.addMenuItemAt()`	Añade un elemento al menú en una ubicación determinada.
`Menu.createMenu()`	Crea una instancia de la clase Menu. Se trata de un método estático.
`Menu.getMenuItemAt()`	Obtiene una referencia a un elemento de menú en una ubicación determinada.
`Menu.hide()`	Cierra un menú.
`Menu.indexOf()`	Devuelve el índice de un elemento de menú determinado.
`Menu.removeAll()`	Elimina todos los elementos de un menú.
`Menu.removeMenuItem()`	Elimina el elemento de menú especificado.
`Menu.removeMenuItemAt()`	Elimina un elemento de un menú en una ubicación determinada.
`Menu.setMenuItemEnabled()`	Indica si un elemento de menú está activado (`true`) o desactivado (`false`).
`Menu.setMenuItemSelected()`	Indica si un elemento de menú está seleccionado (`true`) o no (`false`).
`Menu.show()`	Abre un menú en una ubicación especificada o en una ubicación anterior.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Menu de la clase UIObject. Al llamar a estos métodos desde el objeto Menu, debe utilizarse la forma *`MenuInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.

Método	Descripción
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Menu de la clase UIComponent. Al llamar a estos métodos desde el objeto Menu, debe utilizarse la forma *`MenuInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase Menu

En la siguiente tabla se enumeran las propiedades de la clase Menu.

Propiedad	Descripción
`Menu.dataProvider`	Fuente de datos de un menú.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Menu de la clase UIObject. Al acceder a estas propiedades desde el objeto Menu, debe utilizarse la forma *MenuInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
UIObject.height	Altura del objeto, expresada en píxeles. Sólo lectura.
UIObject.left	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Menu de la clase UIComponent. Al acceder a estas propiedades desde el objeto Menu, debe utilizarse la forma *MenuInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase Menu

En la tabla siguiente se enumeran los eventos de la clase Menu.

Evento	Descripción
`Menu.change`	Se difunde cuando un usuario provoca un cambio en un menú.
`Menu.menuHide`	Se difunde cuando un menú se cierra.
`Menu.menuShow`	Se difunde cuando un menú se abre.
`Menu.rollOut`	Se difunde cuando el puntero deja de estar sobre un elemento.
`Menu.rollOver`	Se difunde cuando el puntero se desplaza sobre un elemento.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Menu de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Menu de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.

Evento	Descripción
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Menu.addMenuItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
menuInstance.addMenuItem(initObject)
```

Sintaxis 2:

```
menuInstance.addMenuItem(childMenuItem)
```

Parámetros

initObject Objeto que contiene propiedades que inicializan los atributos de un elemento de menú. Véase "Atributos XML de elementos de menú" en la página 921.

childMenuItem Objeto de nodo XML.

Valor devuelto

Una referencia al nodo XML añadido.

Descripción

Método; la sintaxis 1 añade un elemento de menú al final del menú. El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade al final del menú un elemento que es un nodo XML creado previamente (con la forma de objeto XML). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

Ejemplo

En el siguiente ejemplo se crean dos menús, inicialmente se añade un elemento de menú a cada uno. A continuación, se añaden dos elementos de menú más al primer menú, llamando a `addMenuItem()` para añadir el primer elemento de menú mediante la especificación de sus atributos. Después, se añade el segundo elemento de menú con el nodo de elemento de menú creado previamente del segundo menú.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear los objetos Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});

// Método de sintaxis 1
first_menu.addMenuItem({label:"Radio Item", instanceName:"radioItem1",
  type:"radio", selected:false, enabled:true, groupName:"myRadioGroup"});

// Método de sintaxis 2
first_menu.addMenuItem(second_menu.getMenuItemAt(0));

// Mostrar menú.
first_menu.show();
```

Menu.addMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

menuInstance.addMenuItemAt(*index, initObject*)

Sintaxis 2:

menuInstance.addMenuItemAt(*index, childMenuItem*)

Parámetros

index Entero que indica la posición de índice (entre los nodos secundarios) en el que se añade el elemento.

initObject Objeto que contiene propiedades que inicializan los atributos de un elemento de menú. Véase "Atributos XML de elementos de menú" en la página 921.

childMenuItem Objeto de nodo XML.

Valor devuelto

Una referencia al nodo XML añadido.

Descripción

Método; la sintaxis 1 añade un elemento (nodo secundario) en una ubicación determinada de un menú. El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade en una ubicación determinada del menú un elemento que es un nodo XML creado previamente (con la forma de objeto XML). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

Ejemplo

En el siguiente ejemplo se crean dos menús, inicialmente se añade un elemento de menú a cada uno. A continuación, se añaden dos elementos de menú más al primer menú, llamando a `addMenuItemAt()` para añadir un elemento de menú en la segunda posición mediante la especificación de sus atributos. Después, se añade un elemento de menú en la tercera posición con el nodo de elemento de menú creado previamente del segundo menú.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear los objetos Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});
```

```
// Método de sintaxis 1
first_menu.addMenuItemAt(1, {label:"Radio Item", instanceName:"radioItem1",
  type:"radio", selected:false, enabled:true, groupName:"myRadioGroup"});

// Método de sintaxis 2
first_menu.addMenuItemAt(2, second_menu.getMenuItemAt(0));

// Mostrar menú.
first_menu.show();
```

Menu.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
menuInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cada vez que el usuario provoca un cambio en el menú.

Los componentes de la versión 2 de la arquitectura de componentes de Macromedia utilizan un modelo de evento distribuidor/detector. Cuando el componente Menu difunde un evento `change`, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Menu.change` tiene además las propiedades adicionales siguientes:

- `menuBar` Referencia a la instancia de MenuBar que es el elemento principal del menú de destino. Si éste no pertenece a una instancia de MenuBar, este valor es `undefined`.
- `menu` Referencia a la instancia de Menu donde está ubicado el elemento de destino.
- `menuItem` Nodo XML, que es el elemento de menú seleccionado.
- `groupName` Cadena que indica el nombre del grupo de botones de opción al que pertenece el elemento. Si el elemento no pertenece a un grupo de botones de opción, este valor es `undefined`.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un menú, `my_menu`, y se define un detector de eventos para él, `menulistener`, que detecta un evento change. Cuando un usuario crea un evento change al hacer clic en un elemento de menú, el ejemplo muestra su atributo de etiqueta en el panel Salida.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
```

```
// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
my_menu.show();

var menuListener:Object = new Object();
menuListener.change = function(evt_obj:Object) {
 trace("Menu item chosen: " + evt_obj.menuItem.attributes.label);
};
my_menu.addEventListener("change", menuListener);
```

Menu.createMenu()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`Menu.createMenu([parent [, mdp]])`

Parámetros

parent Instancia de MovieClip. El clip de película es el componente principal que contiene la nueva instancia de Menu. Este parámetro es opcional.

mdp Instancia de MenuDataProvider que describe esta instancia de Menu. Este parámetro es opcional.

Valor devuelto

Una referencia a la nueva instancia de menú.

Descripción

Método (estático); crea una instancia de Menu y, opcionalmente, la asocia con el elemento principal especificado, con el MenuDataProvider especificado como origen de datos para los elementos de menú.

Si el parámetro *parent* se omite o su valor es null, el menú se asocia con la línea de tiempo `_root`.

Si el parámetro *mdp* se omite o su valor es null, el menú no tiene elementos. Debe efectuar una llamada a los métodos `addMenu()` o `setDataProvider()` para rellenar el menú.

Ejemplo

En el siguiente ejemplo se crea un menú con un submenú para el elemento de menú New. Para crear el menú, se crea un objeto XML, `my_xml`, y se le añaden elementos de menú mediante llamadas a `addMenuItem()`. Después, se crea el menú con una llamada a `createMenu()`, pasando el objeto XML como proveedor de datos.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

var my_xml:XML = new XML();
var newItem_obj:Object = my_xml.addMenuItem({label:"New"});

// Crear otros elementos de submenú para el menú principal.
my_xml.addMenuItem({label:"Open", instanceName:"miOpen"});
my_xml.addMenuItem({label:"Save", instanceName:"miSave"});
my_xml.addMenuItem({type:"separator"});
my_xml.addMenuItem({label:"Quit", instanceName:"miQuit"});

// Crear elementos de submenú para el submenú "New".
newItem_obj.addMenuItem({label:"File..."});
newItem_obj.addMenuItem({label:"Project..."});
newItem_obj.addMenuItem({label:"Resource..."});

// Crear un menú.
var my_menu:Menu = Menu.createMenu(myParent_mc, my_xml);
my_menu.show(100, 20);
```

Menu.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.dataProvider

Descripción

Propiedad; origen de datos de los elementos de un componente Menu.

`Menu.dataProvider` es un objeto de nodo XML. Si se define esta propiedad, se sustituye el origen de datos existente del componente Menu.

El valor predeterminado es `undefined`.

NOTA	Todas las instancias de XML o de XMLNode reciben automáticamente los métodos y las propiedades de la clase MenuDataProvider cuando se utilizan con el componente Menu.

Ejemplo

En el siguiente ejemplo se crea un menú (`my_menu`), se cargan elementos de menú de una página Web en un objeto XML y, a continuación, se llena el menú con elementos de menú mediante la asignación de nodos secundarios del objeto XML a la propiedad `dataProvider` del menú (`my_menu.dataProvider`).

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu();

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean){
 if (success) {
  my_menu.dataProvider = my_xml.firstChild;
 }
}
my_xml.load("http://www.flash-mx.com/mm/xml/menu.xml");

// Mostrar y colocar menús.
my_menu.show(100, 20);
```

Menu.getMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.getMenuItemAt(*index*)

Parámetros

index Entero que indica el índice del nodo en el menú.

Valor devuelto

Una referencia al nodo especificado.

Descripción

Método; devuelve una referencia al nodo secundario especificado del menú.

Ejemplo

En el siguiente ejemplo se crean primero dos menús con un solo elemento de menú para cada uno. A continuación, se añade un segundo elemento de menú al primer menú llamando al método `getmenuItemAt()` para obtener el elemento de menú del segundo menú y añadirlo al primero.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear los objetos Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});

// Añadir elemento desde el segundo menú a la segunda posición del primer
  menú.
first_menu.addMenuItemAt(1, second_menu.getMenuItemAt(0));

// Mostrar menú.
first_menu.show();
```

Menu.hide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.hide()

Valor devuelto

Ninguno.

Descripción

Método; cierra un menú.

Ejemplo

En el siguiente ejemplo se crea un botón y dos menús de elemento y se muestra el menú durante un intervalo de 2000 milisegundos. Cuando caduca el intervalo, la función `closeMenu()` llama al método `menu.hide()` para cerrar el menú. Al hacer clic en el botón Reset Menu se activa el detector `resetMenu()`, que vuelve a mostrar el menú y reinicia el intervalo.

Primero debe arrastrar un componente Menu y un componente Button a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
  - Componente Button en la biblioteca
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10, {label:"Reset Menu",
  _x:100, _y:50});

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();
```

```
// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

my_menu.show(100, 100);
// Llamar a closeMenu después de 2000 milisegundos.
var interval_id:Number = setInterval(closeMenu, 2000, my_menu);
function closeMenu(the_menu:Menu):Void {
  the_menu.hide();
  clearInterval(interval_id);
}
// Detector de clic de botón; mostrar menú y restablecer intervalo.
function resetMenu(evt_obj:Object):Void {
  clearInterval(interval_id);
  my_menu.show(100, 100);
  interval_id = setInterval(closeMenu, 2000, my_menu);
}
my_button.addEventListener("click", resetMenu);
```

Véase también

Menu.show()

Menu.indexOf()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.indexOf(*item*)

Parámetros

item Referencia a un nodo XML que describe un elemento de menú.

Valor devuelto

El índice del elemento de menú especificado, o el valor `undefined` si el elemento no pertenece a este menú.

Descripción

Método; devuelve el índice del elemento de menú especificado en esta instancia de menú.

Ejemplo

En el ejemplo siguiente se crea un menú con dos elementos desde un proveedor de datos XML y, a continuación, se añade un tercer elemento al menú y se guarda la referencia, que es devuelta por el método `addMenuItem()`. Después, se llama al método `indexOf()` utilizando la referencia para obtener el índice del elemento y mostrarlo en el panel Salida.

Primero debe arrastrar un componente Menu y un componente Button a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
var myItem_obj:Object = my_menu.addMenuItem({label:"That item"});

// Mostrar y colocar menús.
my_menu.show(100, 20);

var myIndex_num:Number = my_menu.indexOf(myItem_obj);
trace("Index of 'That Item': " + myIndex_num);
```

Menu.menuHide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.menuHide = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
menuInstance.addEventListener("menuHide", listenerObject);
```

Sintaxis 2:

```
on (menuHide) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cada vez que se cierra un menú.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor-detector. Cuando un componente Menu distribuye un evento `menuHide`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (`listenerObject`), que crea el usuario. Debe llamar al método `addEventListener()` y pasarle el nombre del controlador y el nombre del objeto detector como parámetros.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Menu.menuHide` tiene dos propiedades adicionales:

- `menuBar` Referencia a la instancia de MenuBar que es el elemento principal del menú de destino. Si éste no pertenece a una instancia de MenuBar, este valor es `undefined`.
- `menu` Referencia a la instancia de Menu que está oculta.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un botón y dos menús de elemento. Cuando el usuario hace clic en el botón, un detector del evento de clic de botón muestra el menú. Cuando el usuario hace clic por segunda vez, el menú se oculta y un detector del evento `menuHide`, `menuListener`, muestra "Menu closed" en el panel Salida.

Primero debe arrastrar un componente Menu y un componente Button a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
  - Componente Button en la biblioteca
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10);

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Añadir un botón que muestre el menú cuando se haga clic en el botón.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // obtener referencia al botón
 var the_button:Button = evt_obj.target;
 // Mostrar el menú en la parte inferior del botón.
 my_menu.show(the_button.x, the_button.y + the_button.height);
};
my_button.addEventListener("click", buttonListener);
```

```
// Crear un objeto detector.
var menuListener:Object = new Object();
menuListener.menuHide = function(evt_obj:Object) {
 trace("Menu closed.");
};

// Añadir detector.
my_menu.addEventListener("menuHide", menuListener);
```

Véase también

Menu.menuShow

Menu.menuShow

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.menuShow = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
menuInstance.addEventListener("menuShow", listenerObject);
```

Sintaxis 2:

```
on (menuShow) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cada vez que se abre un menú. Todos los nodos principales abren menús para mostrar los elementos secundarios.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor-detector. Cuando un componente Menu distribuye un evento `menuShow`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (`listenerObject`), que crea el usuario. Debe llamar al método `addEventListener()` y pasarle el nombre del controlador y el objeto detector como parámetros.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Menu.menuShow` tiene dos propiedades adicionales:

- `menuBar` Referencia a la instancia de MenuBar que es el elemento principal del menú de destino. Si éste no pertenece a una instancia de MenuBar, este valor es `undefined`.
- `menu` Referencia a la instancia de Menu que se muestra.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un botón y dos menús de elemento. Cuando el usuario hace clic en el botón, un detector del evento de clic de botón muestra el menú. Un detector del evento`menuShow`, `menuListener`, muestra "Menu open" en el panel Salida.

Primero debe arrastrar un componente Menu y un componente Button a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
  - Componente Button en la biblioteca
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10);

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
```

```
// Añadir un botón que muestre el menú cuando se haga clic en el botón.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // obtener referencia al botón
 var the_button:Button = evt_obj.target;
 // Mostrar el menú en la parte inferior del botón.
 my_menu.show(the_button.x, the_button.y + the_button.height);
};
my_button.addEventListener("click", buttonListener);

// Crear un objeto detector.
var menuListener:Object = new Object();
menuListener.menuShow = function(evt_obj:Object) {
 trace("Menu open.");
};

// Añadir detector.
my_menu.addEventListener("menuShow", menuListener);
```

Véase también

Menu.menuHide

Menu.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.removeAll()

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos y actualiza el menú.

Ejemplo

En el ejemplo siguiente se crea un menú con dos elementos y, después de un intervalo de un par de segundos (2000 milisegundos), se eliminan todos los nodos del menú.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);
var interval_id:Number = setInterval(remove, 2000, my_menu);
function remove(the_menu:Menu):Void {
  // Borrar todos los elementos de menú.
  the_menu.removeAll();
  clearInterval(interval_id);
  the_menu.show(100, 20);
}
```

Menu.removeMenuItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.removeMenuItem()

Valor devuelto

Una referencia al elemento de menú que se devuelve (nodo XML). Si no hay ningún elemento en esta posición, el valor es `undefined`.

Descripción

Método; elimina el elemento de menú especificado y todos sus elementos secundarios. Además, actualiza el menú.

Ejemplo

En el siguiente ejemplo se crea un menú con tres elementos de menú y se establece un intervalo para hacer que el menú se muestre durante un par de segundos (2000 milisegundos). Cuando caduca el intervalo, el ejemplo llama a la función `removeItem()`, que llama al método `removeMenuItem()` para eliminar el primer elemento del menú y volver a mostrarlo.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");
```

```
// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"first Item"});
menuDP_obj.addMenuItem({label:"second Item"});
menuDP_obj.addMenuItem({label:"third Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar el menú.
my_menu.show(100, 20);

// Llamar a closeMenu después de 2000 milisegundos.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Borrar el primer elemento de nodo.
  var myItem_obj:Object = my_menu.getMenuItemAt(0);
  myItem_obj.removeMenuItem();
  clearInterval(interval_id);
  my_menu.show(100, 20);
}
```

Menu.removeMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.removeMenuItemAt(*index*)

Parámetros

index Índice del elemento de menú que se va a eliminar.

Valor devuelto

Una referencia al elemento de menú que se devuelve (nodo XML). Si no hay ningún elemento en esta posición, el valor es `undefined`.

Descripción

Método; elimina un elemento de menú y todos sus elementos secundarios en el índice especificado. Si no hay ningún elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno.

Ejemplo

En el ejemplo siguiente se crea un menú con dos elementos y, después de un intervalo de un par de segundos (2000 milisegundos), se elimina el segundo elemento (en el índice 1).

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);
var interval_id:Number = setInterval(remove, 2000, my_menu);
function remove(the_menu:Menu):Void {
  // Borrar el segundo elemento de nodo.
  var item_obj:Object = my_menu.removeMenuItemAt(1);
  trace("Item removed: " + item_obj);
  clearInterval(interval_id);
  the_menu.show(100, 20);
}
```

Menu.rollOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.rollOut = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
menuInstance.addEventListener("rollOut", listenerObject);
```

Sintaxis 2:

```
on (rollOut) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero deja de estar sobre un elemento de menú.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor-detector. Cuando un componente Menu difunde un evento `rollOut`, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Menu.rollOut` tiene una propiedad adicional: `menuItem`, que es una referencia al elemento de menú (nodo XML) que el puntero ha dejado de señalar.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un menú con dos elementos y un detector para un evento `rollOut`. Cuando se difunde el evento `rollOut`, una función `trace`() del controlador de eventos, `menuListener`, muestra el nombre del elemento de menú para el que se produce el evento.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);

// Crear un objeto detector.
var menuListener:Object = new Object();
menuListener.rollOut = function(evt_obj:Object) {
 trace("Menu rollOut: " + evt_obj.menuItem.attributes.label);
};

// Añadir detector.
my_menu.addEventListener("rollOut", menuListener);
```

Menu.rollOver

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.rollOver = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
menuInstance.addEventListener("rollOver", listenerObject);
```

Sintaxis 2:

```
on (rollOver) {
  // Introducir aquí el código propio.
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza sobre un elemento de menú.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor-detector. Cuando un componente Menu difunde un evento `rollover`, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Menu.rollOver` tiene una propiedad adicional: `menuItem`, que es una referencia al elemento de menú (nodo XML) sobre el que se ha desplazado el puntero.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un menú con dos elementos y un detector para un evento `rollOver`. Cuando se difunde el evento `rollOver`, una función `trace`() del controlador de eventos, `menuListener`, muestra el nombre del elemento de menú para el que se produce el evento.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);

// Crear un objeto detector.
var menuListener:Object = new Object();
menuListener.rollOver = function(evt_obj:Object) {
 trace("Menu rollOver: "+evt_obj.menuItem.attributes.label);
};

// Añadir detector.
my_menu.addEventListener("rollOver", menuListener);
```

Menu.setMenuItemEnabled()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.setMenuItemEnabled(*item*, *enable*)

Parámetros

item Nodo XML; el nodo del elemento de menú de destino en el proveedor de datos.

enable Valor booleano que indica si el elemento está activado (`true`) o desactivado (`false`).

Valor devuelto

Ninguno.

Descripción

Método; cambia el atributo `enabled` del elemento de destino por el estado especificado en el parámetro *enable*. Si esta llamada provoca un cambio de estado, el elemento se vuelve a dibujar con el estado nuevo.

Ejemplo

En el siguiente ejemplo se crea un menú con dos elementos de menú y se llama al método `setMenuItemEnabled()` para desactivar el primero.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();
```

```
// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Seleccionar el primer elemento de menú y desactivarlo.
var item_obj:Object = my_menu.getMenuItemAt(0);
my_menu.setMenuItemEnabled(item_obj, false);

// Mostrar y colocar el menú.
my_menu.show(100, 20);
```

Véase también

Menu.setMenuItemSelected()

Menu.setMenuItemSelected()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.setMenuItemSelected(*item*, *select*)

Parámetros

item Nodo XML. El nodo del elemento del menú destino en el proveedor de datos.

select Valor booleano que indica si el elemento está seleccionado (true) o no lo está (false). Si el elemento es una casilla de verificación, este valor indica si la marca está visible o no. Si el elemento es un botón de opción, pasa a ser la selección actual del grupo de opciones.

Valor devuelto

Ninguno.

Descripción

Método; cambia el atributo `selected` del elemento por el estado especificado por el parámetro *`select`*. Si esta llamada provoca un cambio de estado, el elemento se vuelve a dibujar con el estado nuevo. Sólo es aplicable a los elementos cuyo atributo `type` sea `"radio"` o `"check"`, porque hace aparecer o desaparecer su punto o su marca de verificación. Una llamada a este método en un elemento cuyo tipo sea `"normal"` o `"separator"` no tendrá ningún efecto.

Ejemplo

En el siguiente ejemplo se crea un menú con dos elementos de menú, siendo el segundo de ellos un elemento de menú de casilla de verificación. El ejemplo llama al método `setMenuItemSelected()` para poner el elemento de menú de casilla de verificación en un estado seleccionado.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({type:"check", label:"2nd Item"})

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

var myItem = my_menu.getMenuItemAt(1);
my_menu.setMenuItemSelected(myItem, true);

// Mostrar y colocar el menú.
my_menu.show(100, 20);
```

Menu.show()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

menuInstance.show(*x*, *y*)

Parámetros

x Coordenada *x*.

y Coordenada *y*.

Valor devuelto

Ninguno.

Descripción

Método; abre un menú en una ubicación determinada. El menú cambia su tamaño automáticamente para que se puedan ver todos los elementos del nivel superior. La esquina superior izquierda se coloca en la ubicación especificada del sistema de coordenadas que proporciona el elemento principal del componente.

Si los parámetros *x* e *y* se omiten, el menú aparece en la ubicación anterior.

Ejemplo

En el siguiente ejemplo se crea un menú de un objeto de menú XML y se llama al método `menu.show()` para que lo muestre.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

var my_xml:XML = new XML();
```

```
// Crear elementos para el menú.
var newItem_obj:Object = my_xml.addMenuItem({label:"New"});
my_xml.addMenuItem({label:"Open", instanceName:"miOpen"});
my_xml.addMenuItem({label:"Save", instanceName:"miSave"});
my_xml.addMenuItem({type:"separator"});
my_xml.addMenuItem({label:"Quit", instanceName:"miQuit"});

// Crear y mostrar el menú.
var my_menu:Menu = Menu.createMenu(myParent_mc, my_xml);
my_menu.show(100, 20);
```

Véase también

`Menu.hide()`

Clase MenuDataProvider

Nombre de clase de ActionScript mx.controls.menuclasses.MenuDataProvider

La clase MenuDataProvider es una clase de "decoración" (adición) que añade funcionalidad a la clase global XMLNode. Esta funcionalidad permite que instancias de XML asignadas a una propiedad `Menu.dataProvider` utilicen los métodos y propiedades de MenuDataProvider para manipular sus propios datos, así como las vistas de menú asociadas.

Recuerde los conceptos siguientes, relacionados con la clase MenuDataProvider:

- MenuDataProvider es una clase de "decoración" (adición). No es necesario crear una instancia para usarla.
- Los menús aceptan directamente XML como valor de la propiedad `dataProvider`.
- Si se crea una instancia de una clase Menu, todas las instancias XML del archivo SWF se "decoran" con la clase MenuDataProvider.
- Sólo los métodos de MenuDataProvider difunden eventos a los componentes Menu. Puede seguir utilizando métodos XML nativos, pero éstos no difunden eventos que actualizan las vistas de Menu. Para controlar el modelo de datos, use los métodos MenuDataProvider. Para operaciones de sólo lectura, como el desplazamiento por la jerarquía de Menu, utilice los métodos XML.
- Todos los elementos del componente Menu son objetos XML "decorados" con la clase MenuDataProvider.
- Los cambios efectuados en los atributos del elemento no se reflejan en el menú en pantalla hasta que se vuelva a dibujar.

Resumen de métodos de la clase MenuDataProvider

En la tabla siguiente se enumeran los métodos de la clase MenuDataProvider.

Método	Descripción
`MenuDataProvider.addMenuItem()`	Añade un elemento secundario.
`MenuDataProvider.addMenuItemAt()`	Añade un elemento secundario en una ubicación determinada.
`MenuDataProvider.getMenuItemAt()`	Obtiene una referencia a un elemento de menú en una ubicación determinada.
`MenuDataProvider.indexOf()`	Devuelve el índice de un elemento de menú determinado.
`MenuDataProvider.removeMenuItem()`	Elimina un elemento de menú.
`MenuDataProvider.removeMenuItemAt()`	Elimina un elemento de menú de una ubicación determinada.

MenuDataProvider.addMenuItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
myMenuDataProvider.addMenuItem(initObject)
```

Sintaxis 2:

```
myMenuDataProvider.addMenuItem(childMenuItem)
```

Parámetros

`initObject` Objeto que contiene los atributos que inicializan los atributos de un elemento del componente Menu. Para más información, consulte "Atributos XML de elementos de menú" en la página 921.

`childMenuItem` Nodo XML.

Valor devuelto

Una referencia a un objeto XMLNode.

Descripción

Método; la sintaxis 1 añade un elemento secundario al final de un elemento de menú principal (que puede ser el propio menú). El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade al final de un elemento de menú principal un elemento secundario que está definido en el parámetro XML *childMenuItem* especificado.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se crea un menú desde un proveedor de datos XML. Se llama al método `addMenuItem()` para añadir dos elementos al menú principal y, además, añadir dos elementos a un submenú para el primer elemento del menú principal.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");


// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"Folders"});
menuDP_obj.addMenuItem({label:"Radio Edit", type:"radio"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
```

```
// Mostrar y colocar el menú.
my_menu.show(100, 20);

// Recuperar el primer elemento del menú y añadirle elementos.
var item_obj:Object = menuDP_obj.getMenuItemAt(0);
item_obj.addMenuItem({label:"First item", instanceName:"firstItem1"});
item_obj.addMenuItem({label:"Second item", instanceName:"secondItem1"});
```

MenuDataProvider.addMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

myMenuDataProvider.addMenuItemAt(*index*, *initObject*)

Sintaxis 2:

myMenuDataProvider.addMenuItemAt(*index*, *childMenuItem*)

Parámetros

index Entero.

initObject Objeto que contiene los atributos específicos que inicializan los atributos de un elemento de menú. Para más información, consulte "Atributos XML de elementos de menú" en la página 921.

childMenuItem Nodo XML.

Valor devuelto

Una referencia al nodo XML añadido.

Descripción

Método; la sintaxis 1 añade un elemento secundario en la posición de índice especificada en el elemento de menú principal (que puede ser el propio menú). El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade un elemento secundario que está definido en el parámetro XML *childMenuItem* especificado en el índice determinado de un elemento de menú principal.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se crea un menú con un elemento de menú y, a continuación, se llama al método `addMenuItemAt()` para añadir un segundo elemento.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"Edit"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar el menú.
my_menu.show(100, 20);

// Añadir el elemento de menú.
menuDP_obj.addMenuItemAt(1, {label:"Save", instanceName:"saveItem1"});
```

MenuDataProvider.getMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myMenuDataProvider.getMenuItemAt(*index*)

Parámetros

index Entero que indica la posición del menú.

Valor devuelto

Una referencia al nodo XML especificado.

Descripción

Método; devuelve una referencia al elemento de menú secundario especificado del elemento de menú actual.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se crea un menú, se le añade un elemento de menú y, a continuación, se llama al método `getMenuItemAt()` para acceder a su objeto de nodo con el fin de añadirle un elemento de submenú. También se llama al método `getMenuItemAt()` para mostrar la etiqueta del elemento de submenú en el panel Salida.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
var menuItem_obj:Object = menuDP_obj.getMenuItemAt(0);
menuItem_obj.addMenuItem({label:"Submenu Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
```

```
// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);

// Recuperar el elemento de submenú del primer elemento de menú.
var myMenuItem_obj:Object = menuDP_obj.firstChild;
trace(myMenuItem_obj.getMenuItemAt(0));
```

MenuDataProvider.indexOf()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myMenuDataProvider.indexOf(*item*)

Parámetros

item Referencia al nodo XML que describe el elemento de menú.

Valor devuelto

El índice del elemento de menú especificado; devuelve el valor `undefined` si el elemento no pertenece a este menú.

Descripción

Método; devuelve el índice del elemento de menú especificado en este elemento de menú principal.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se añade un elemento de menú a un menú y se llama al método `indexOf()` para mostrar el índice del elemento en el panel Salida.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);

// Añadir un elemento y realizar un seguimiento de la posición de ese
  elemento.
var myItem_obj:Object = menuDP_obj.addMenuItem({label:"That item"});
var myIndex_num:Number = menuDP_obj.indexOf(myItem_obj);
trace("Position: " + myIndex_num);
```

MenuDataProvider.removeMenuItem()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myMenuDataProvider.removeMenuItem()

Valor devuelto

Una referencia al elemento de menú eliminado (nodo XML); se devuelve el valor `undefined` si se produce un error.

Descripción

Método; elimina el elemento de destino y sus nodos secundarios.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se crea un menú con tres elementos de menú y, después de un intervalo de un par de segundos (2000 milisegundos) se llama a `removeMenuItem()` para eliminar el primer elemento de menú.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();
d
// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
menuDP_obj.addMenuItem({label:"3rd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);
// Llamar a removeItem después de 2000 milisegundos.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Eliminar el elemento en la posición 0.
  var myItem_obj:Object = menuDP_obj.getMenuItemAt(0);
  myItem_obj.removeMenuItem();
  clearInterval(interval_id);
}
```

MenuDataProvider.removeMenuItemAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myMenuDataProvider.removeMenuItemAt(*index*)

Parámetros

index Índice del elemento de menú.

Valor devuelto

Una referencia al elemento de menú eliminado. Si no hay ningún elemento en esta posición, el valor es `undefined`.

Descripción

Método; elimina el elemento secundario del elemento de menú especificado en el parámetro *index*. Si no hay ningún elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno.

Los nodos o los elementos de menú de una instancia de MenuDataProvider pueden llamar a los métodos de la clase MenuDataProvider.

Ejemplo

En el siguiente ejemplo se crea un menú con tres elementos de menú y, después de un intervalo de un par de segundos (2000 milisegundos), se llama a `removeMenuItemAt()` para eliminar el primer elemento de menú.

Primero debe arrastrar un componente Menu a la biblioteca y, después, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Menu en la biblioteca
*/

import mx.controls.Menu;

// Crear un objeto XML para que actúe como un almacén.
var my_xml:XML = new XML();
```

```
// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método createMenu(), que figura a continuación, espera
// recibir un elemento raíz cuyos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Añadir elementos de menú.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
menuDP_obj.addMenuItem({label:"3rd Item"});

// Crear el objeto Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Mostrar y colocar menús.
my_menu.show(100, 20);
// Llamar a removeItem después de 2000 milisegundos.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Eliminar el elemento en la posición 0.
  menuDP_obj.removeMenuItemAt(0);
  clearInterval(interval_id);
}
```

CAPÍTULO 31

Componente MenuBar (sólo en Flash Professional)

31

El componente MenuBar sirve para crear una barra de menús horizontal con menús emergentes y comandos, como las barras de menús que contienen los menús Archivo y Edición de la mayoría de las aplicaciones de software. El componente MenuBar complementa el componente Menu; proporciona una interfaz, en la que se puede hacer clic, que muestra y oculta menús que se comportan como un grupo para la interactividad del ratón y del teclado.

El componente MenuBar permite crear con pocos pasos un menú de aplicación. Para crear una barra de menús, puede asignar un proveedor de datos XML a la barra de menús que describa una serie de menús, o bien puede utilizar el método `MenuBar.addMenu()` para añadir instancias de menú de una en una.

Cada uno de los menús de la barra de menús consta de dos partes: el menú y el botón que hace que el menú se abra (denominado activador de menú). Estos activadores de menú en los que se hace clic aparecen en la barra de menús como una etiqueta de texto, cuyo estado de resaltado de borde puede ser saliente o entrante; dichos estados reaccionan ante interacciones del ratón y del teclado.

Cuando se hace clic en un activador de menú, el menú correspondiente se abre por debajo de éste. El menú permanecerá activo hasta se vuelva a hacer clic en el activador, o bien hasta que se seleccione un elemento de menú o se haga clic fuera del área del menú.

Además de crear activadores de menú que muestran u ocultan menús, el componente MenuBar crea un comportamiento de grupo entre una serie de menús. Ello permite al usuario explorar un gran número de opciones de comando desplazándose por la serie de activadores o utilizando las teclas de flecha para desplazarse por las listas. El ratón y el teclado funcionan conjuntamente de forma interactiva para que el usuario pueda desplazarse por los menús de la barra de menús.

Un usuario no puede desplazarse por los menús de una barra de menús. Si la anchura de los menús es superior a la anchura de la barra de menús, éstos se enmascaran.

Los lectores de pantalla no pueden acceder al componente Menubar.

Los menús a menudo se anidan en barras de menú. Para más información sobre menús, consulte “Componente Menu (sólo en Flash Professional)” en la página 917.

Interacción con el componente MenuBar (sólo en Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente MenuBar.

Si desplaza el ratón por un activador de menú se muestra un resaltado de borde externo alrededor de la etiqueta del activador.

Si se selecciona una instancia de MenuBar haciendo clic en ésta o utilizando el tabulador, podrá utilizar las teclas siguientes para controlarla:

Tecla	Descripción
Flecha abajo	Desplaza la selección a la fila inferior siguiente del menú.
Flecha arriba	Desplaza la selección a la fila superior siguiente del menú.
Flecha derecha	Desplaza la selección al botón siguiente.
Flecha izquierda	Desplaza la selección al botón anterior.
Intro/Escape	Cierra un menú abierto.

NOTA

Si un menú está abierto, no se puede presionar el tabulador para cerrarlo. Debe realizar una selección o cerrar el menú con la tecla Esc.

Utilización del componente MenuBar (sólo en Flash Professional)

Puede utilizar el componente MenuBar para añadir un conjunto de menús (por ejemplo, Archivo, Edición, Especial o Ventana) al borde superior de una aplicación.

Parámetros de MenuBar

A continuación se indica el parámetro de edición que se puede definir para cada instancia del componente MenuBar en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

Labels Matriz que añade activadores de menú con las etiquetas especificadas al componente MenuBar. El valor predeterminado es `[]` (matriz vacía).

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente MenuBar en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en `UIObject` y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

No se puede acceder al parámetro Labels utilizando código ActionScript. No obstante, se puede escribir código ActionScript para controlar opciones adicionales del componente MenuBar mediante sus propiedades, métodos y eventos. Para más información, consulte "Clase MenuBar (sólo en Flash Professional)" en la página 987.

Creación de aplicaciones con el componente MenuBar

En este ejemplo, se arrastra un componente MenuBar al escenario, se añade código para añadirle elementos de menú y se asocian detectores al menú para responder a la selección de un elemento de menú.

Para utilizar un componente MenuBar en una aplicación:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Arrastre el componente MenuBar desde el panel Componentes al escenario.
3. Coloque el menú en la parte superior del escenario si desea un diseño estándar.
4. Seleccione la instancia de MenuBar y, en el inspector de propiedades, introduzca el nombre de instancia **my_mb**.
5. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;

var my_menu:Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});
my_menu.addMenuItem({label:"Close", instanceName:"closeInstance"});
```

Con este código se añade un menú File a la instancia de MenuBar. Después, se utiliza un método Menu para añadir tres elementos de menú: New, Open y Close.

6. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
//Crear un objeto detector.
var mbListener:Object = new Object();
mbListener.change = function(evt_obj:Object) {
 var menuItem_obj:Object = evt_obj.menuItem;
 switch (menuItem_obj.attributes.instanceName) {
 case "newInstance":
  trace("New menu item");
  break;
 case "openInstance":
  trace("Open menu item");
  break;
 case "closeInstance":
  trace("Close menu item");
  break;
 }
 trace(menuItem_obj);
};

//Añadir detector.
my_menu.addEventListener("change", mbListener);
```

Este código crea un objeto detector, `mblistener`, que captura una selección de elemento de menú y muestra su nombre y el valor del objeto de elemento de menú.

NOTA	debe llamar al método `addEventListener()` para registrar el detector con la instancia de menú, no con la instancia de la barra de menús.

7. Seleccione Control > Probar película para probar el componente MenuBar.

Personalización del componente MenuBar (sólo en Flash Professional)

Este componente adopta el tamaño adecuado en función de las etiquetas de los activadores que se proporcionan en la propiedad `dataProvider` o los métodos de la clase MenuBar. Cuando hay un botón de activador en una barra de menús, éste se mantiene con un tamaño fijo que depende de los estilos de fuente y de la longitud del texto.

Utilización de estilos con el componente MenuBar

El componente MenuBar crea una etiqueta de activador para cada uno de los menús de un grupo. Utilice los estilos para cambiar el aspecto de las etiquetas de los activadores. El componente MenuBar admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

El componente MenuBar también reenvía todos los valores de las propiedades de estilo del componente Menu a las instancias compuestas de éste. Para obtener una lista de propiedades de estilo del componente Menu, consulte “Utilización de estilos con el componente Menu” en la página 931.

Utilización de aspectos con el componente MenuBar

El componente MenuBar utiliza tres aspectos para representar su fondo, utiliza un símbolo de clip de película para resaltar elementos individuales y contiene un componente Menu como menú emergente que puede utilizar aspectos. En la siguiente tabla se detallan los aspectos de MenuBar. Para más información sobre la aplicación de aspectos en el componente Menu, consulte "Utilización de aspectos con el componente Menu" en la página 935.

El componente MenuBar admite las siguientes propiedades de aspecto:

Propiedad	Descripción
`menuBarBackLeftName`	Estado sin presionar del icono emergente
`menuBarBackRightName`	Estado presionado del icono emergente
`menuBarBackMiddleName`	Estado desactivado del icono emergente

Para crear símbolos de clip de película para aspectos de MenuBar:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta MenuBar Assets a la biblioteca del documento.
4. Expanda la carpeta MenuBar Assets/Elements en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo MenuBarBackLeft.
6. Personalice el símbolo como desee.

   Por ejemplo, deje el borde exterior en blanco.
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.

   Por ejemplo, defina los bordes exteriores de los símbolos central y derecho con color negro.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente MenuBar al escenario.
10. Defina las propiedades de MenuBar para que muestren los elementos en la barra.

11. Seleccione Control > Probar película.

NOTA	El borde utilizado para resaltar elementos individuales de un componente MenuBar es una instancia de ActivatorSkin, que se encuentra en la carpeta Flash UI Components 2/Themes/MMDefault/Button Assets. Se puede personalizar este símbolo para que apunte a una clase diferente y proporcione otro borde. Sin embargo, el nombre del símbolo no se puede modificar y tampoco se puede utilizar un símbolo diferente para instancias de MenuBar distintas de un mismo documento.

Clase MenuBar (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > MenuBar

Nombre de clase de ActionScript mx.controls.MenuBar

Los métodos y las propiedades de la clase MenuBar permiten crear una barra de menús horizontal con menús emergentes y comandos. Estos métodos y propiedades complementan los de la clase Menu; permiten crear una interfaz, en la que se puede hacer clic, que muestra y oculta menús que se comportan como un grupo para la interactividad del ratón y del teclado.

Resumen de métodos de la clase MenuBar

En la tabla siguiente se enumeran los métodos de la clase MenuBar.

Método	Descripción
`MenuBar.addMenu()`	Añade un elemento a la barra de menús.
`MenuBar.addMenuAt()`	Añade un menú de una ubicación determinada a la barra de menús.
`MenuBar.getMenuAt()`	Obtiene una referencia a un menú en una ubicación determinada.
`MenuBar.getMenuEnabledAt()`	Devuelve un valor booleano que indica si el menú está activado (`true`) o desactivado (`false`).
`MenuBar.removeMenuAt()`	Elimina, de una barra de menús, un menú de una ubicación determinada.
`MenuBar.removeAll()`	Elimina todos los elementos de menú de la barra de menús.
`MenuBar.setMenuEnabledAt()`	Valor booleano que indica si un menú puede elegirse (`true`) o no (`false`).

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase MenuBar de la clase UIObject. Al llamar a estos métodos desde el objeto MenuBar, debe utilizarse la forma `MenuBar.methodName`.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase MenuBar de la clase UIComponent. Al llamar a estos métodos desde el objeto MenuBar, debe utilizarse la forma `MenuBar.methodName`.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase MenuBar

En la tabla siguiente se enumeran las propiedades de la clase MenuBar.

Propiedad	Descripción
`MenuBar.dataProvider`	Modelo de datos de una barra de menús.
`MenuBar.labelField`	Cadena que determina qué atributo de cada XMLNode se va a utilizar como texto de etiqueta del menú.
`MenuBar.labelFunction`	Función que determina lo que se debe mostrar en cada etiqueta de menú.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase MenuBar de la clase UIObject. Al llamar a estas propiedades desde el objeto MenuBar, debe utilizarse la forma `MenuBar.`*`propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase MenuBar de la clase UIComponent. Al llamar a estas propiedades desde el objeto MenuBar, debe utilizarse la forma `MenuBar.propertyName`.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase MenuBar

No hay eventos exclusivos de la clase MenuBar.

Eventos heredados de la clase Menu

En la tabla siguiente se enumeran los eventos que hereda la clase MenuBar de la clase Menu. Al llamar a estos eventos desde el objeto MenuBar, debe utilizarse la forma `MenuBar.eventName`.

Evento	Descripción
`Menu.change`	Se difunde cuando un usuario provoca un cambio en un menú.
`Menu.menuHide`	Se difunde cuando un menú se cierra.
`Menu.menuShow`	Se difunde cuando un menú se abre.
`Menu.rollOut`	Se difunde cuando el puntero deja de estar sobre un elemento.
`Menu.rollOver`	Se difunde cuando el puntero se desplaza sobre un elemento.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase MenuBar de la clase UIObject. Al llamar a estos eventos desde el objeto MenuBar, debe utilizarse la forma `MenuBar.eventName`.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase MenuBar de la clase UIComponent. Al llamar a estos eventos desde el objeto MenuBar, debe utilizarse la forma `MenuBar.eventName`.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

MenuBar.addMenu()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
menuBarInstance.addMenu(label)
```

Sintaxis 2:

```
menuBarInstance.addMenu(label, menuDataProvider)
```

Parámetros

label Cadena que indica la etiqueta del nuevo menú.

menuDataProvider Instancia de XML o de XMLNode que describe el menú y sus elementos. Si el valor es una instancia de XML, se utiliza el primer elemento secundario de la instancia.

Valor devuelto

Una referencia al nuevo objeto de menú.

Descripción

Método; la sintaxis 1 añade un menú único y un activador de menú al final de la barra de menús y utiliza la etiqueta especificada. La sintaxis 2 añade un menú único y un activador de menús definidos en el parámetro *menuDataProvider* XML especificado.

Ejemplo

Sintaxis 1: En el ejemplo siguiente se añade un menú File y, a continuación, se utiliza `Menu.addMenuItem()` para añadir los elementos de menú New y Open.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/
var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});
```

Sintaxis 2: En el ejemplo siguiente se añade un menú Font con los elementos de menú Bold e Italic, que están definidos en el proveedor de datos XML `myDP_xml`:

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.addMenuItem({type:"check", label:"Bold", instanceName:"check1"});
myDP_xml.addMenuItem({type:"check", label:"Italic",
  instanceName:"check2"});

var my_menu:mx.controls.Menu = my_mb.addMenu("Font", myDP_xml);
```

MenuBar.addMenuAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
menuBarInstance.addMenuAt(index, label)
```

Sintaxis 2:

```
menuBarInstance.addMenuAt(index, label, menuDataProvider)
```

Parámetros

index Entero que indica la posición en la que debe insertarse el menú. La primera posición es 0. Para realizar la adición al final del menú, llame al método `MenuBar.addMenu(label)`.

label Cadena que indica la etiqueta del nuevo menú.

menuDataProvider Instancia de XML o de XMLNode que describe el menú. Si el valor es una instancia de XML, se utiliza el primer elemento secundario de la instancia.

Valor devuelto

Una referencia al nuevo objeto de menú.

Descripción

Método; la sintaxis 1 añade un único menú y un activador de menú en el índice especificado con la etiqueta determinada. La sintaxis 2 añade un menú único y un activador de menú con etiqueta en el índice especificado. El contenido del menú se define en el parámetro *menuDataProvider*.

Ejemplo

Sintaxis 1: En el siguiente ejemplo se coloca un menú en la primera posición de la instancia de MenuBar `my_mb`.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenuAt(0, "Flash");
my_menu.addMenuItem({label:"About Macromedia Flash",
  instanceName:"aboutInst"});
my_menu.addMenuItem({label:"Preferences", instanceName:"PrefInst"});
```

Sintaxis 2: En el ejemplo siguiente se añade un menú Edit con los elementos de menú Undo, Redo, Cut y Copy, que están definidos en el proveedor de datos XML `myDP_xml`. Se añade el menú a la primera posición de la instancia de MenuBar `my_mb`.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.addMenuItem({label:"Undo", instanceName:"undoInst"});
myDP_xml.addMenuItem({label:"Redo", instanceName:"redoInst"});
myDP_xml.addMenuItem({type:"separator"});
myDP_xml.addMenuItem({label:"Cut", instanceName:"cutInst"});
myDP_xml.addMenuItem({label:"Copy", instanceName:"copyInst"});

my_mb.addMenuAt(0, "Edit", myDP_xml);
```

MenuBar.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
menuBarInstance.dataProvider
```

Descripción

Propiedad; modelo de datos de los elementos de un componente MenuBar.

`MenuBar.dataProvider` es un objeto de nodo XML. Si se define esta propiedad, se sustituye el modelo de datos existente del componente MenuBar. Los nodos secundarios que tenga el proveedor de datos se utilizan como elementos de la propia barra de menús; los subnodos de estos nodos secundarios se utilizan como elementos de sus respectivos menús.

El valor predeterminado es `undefined`.

NOTA	Todas las instancias de XML o de XMLNode reciben automáticamente los métodos y las propiedades de la clase MenuDataProvider cuando se utilizan con el componente MenuBar.

Ejemplo

En el siguiente ejemplo se carga un archivo de menú XML desde una página Web y se utiliza el controlador de eventos `onLoad` para asignarlo a la propiedad `dataProvider` de la instancia de MenuBar `my_mb`.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.ignoreWhite = true;
myDP_xml.onLoad = function(success:Boolean) {
 if (success) {
  my_mb.dataProvider = myDP_xml.firstChild;
 } else {
  trace("error loading XML file");
 }
};
myDP_xml.load("http://www.flash-mx.com/mm/xml/menubar.xml");
```

MenuBar.getMenuAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`menuBarInstance.getMenuAt(`*`index`*`)`

Parámetros

index Entero que indica la posición del menú.

Valor devuelto

Una referencia al menú en el índice especificado. Si no hay ningún menú en esta posición, el valor es `undefined`.

Descripción

Método; devuelve una referencia al menú en el índice especificado. Puesto que `getMenuAt()` devuelve una referencia, es posible añadir elementos a un menú en el índice especificado.

Ejemplo

En el siguiente ejemplo se crea un menú File y se llama a `getMenuAt()`, que crea una referencia para él. A continuación, se utiliza la referencia para añadir dos elementos de menú, New y Open, al menú File.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

my_mb.addMenu("File");

var my_menu:mx.controls.Menu = my_mb.getMenuAt(0);
my_menu.addMenuItem({label:"New",instanceName:"newInst"});
my_menu.addMenuItem({label:"Open",instanceName:"openInst"});
```

MenuBar.getMenuEnabledAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`menuBarInstance.getMenuEnabledAt(`*`index`*`)`

Parámetros

`index` Índice del menú en la barra de menús.

Valor devuelto

Valor booleano que indica si este menú puede elegirse (`true`) o no (`false`).

Descripción

Método; devuelve un valor booleano que indica si este menú puede elegirse (`true`) o no (`false`).

Ejemplo

En el siguiente ejemplo se crea un menú File con dos elementos de menú y, a continuación, se llama a `setMenuEnabledAt()` con un valor de `false` para desactivarlo. También se llama a `getMenuEnabledAt()` y se muestra el resultado para enseñar al usuario cómo determinar si un menú está activado.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

//Desactivar menú "file".
my_mb.setMenuEnabledAt(0, false);

//Comprobar si se puede seleccionar el menú "file".
trace("Menu can be selected: " + my_mb.getMenuEnabledAt(0));
```

MenuBar.labelField

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
menuBarInstance.labelField
```

Descripción

Propiedad; cadena que especifica el atributo de cada nodo XML que se va a utilizar como texto de etiqueta del menú. El valor de esta propiedad también se pasa a los menús que se crean desde la barra de menús. El valor predeterminado es `"label"`.

Después de definir la propiedad `dataProvider`, esta propiedad es de sólo lectura.

Ejemplo

En el siguiente ejemplo se especifica que el atributo `name` de cada nodo XML proporcionará el texto de etiqueta de los elementos de menú.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/
var my_mb:mx.controls.MenuBar;

//Cambiar texto de etiqueta para que se lea desde "name".
my_mb.labelField = "name";

var my_menu:mx.controls.Menu = my_mb.addMenu({name:"File"});
my_menu.addMenuItem({name:"New", instanceName:"newInstance"});
my_menu.addMenuItem({name:"Open", instanceName:"openInstance"});
```

MenuBar.labelFunction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
menuBarInstance.labelFunction
```

Descripción

Propiedad; función que determina lo que se debe mostrar en cada texto de etiqueta de menú. La función acepta el nodo XML asociado con un elemento como parámetro y devuelve la cadena que se va a utilizar como texto de etiqueta. Esta propiedad se pasa a los menús que se crean desde la barra de menús. El valor predeterminado es `undefined`.

Después de definir la propiedad `dataProvider`, esta propiedad es de sólo lectura.

Ejemplo

En el siguiente ejemplo se utiliza una función de etiqueta para crear y devolver una etiqueta personalizada, como por ejemplo New (Control +N), desde los atributos de nodo.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", data:"Control+N",
  instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", data:"Control+O",
  instanceName:"openInstance"});
my_menu.addMenuItem({label:"Close", data:"Control+W",
  instanceName:"closeInstance"});

//Dar formato a los datos XML proporcionados para el menú.
my_menu.labelFunction = function(node:XMLNode):String  {
 var attrb:Object = node.attributes;
 return (attrb.label + " (" + attrb.data + ")");
};
```

MenuBar.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
menuBarInstance.removeAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de menú de la barra de menús.

Ejemplo

En el siguiente ejemplo se crean los menús File, Edit, Tools y Window en la barra de menús. Después, cuando se hace clic en un botón, el script llama a `removeAll()` para eliminar los elementos de menú.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `myMenuBar` en el inspector de propiedades. Arrastre también el componente Button al escenario e introduzca el nombre de instancia `remBtn`. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var menu = myMenuBar.addMenu("File");
var menu = myMenuBar.addMenu("Edit");
var menu = myMenuBar.addMenu("Tools");
var menu = myMenuBar.addMenu("Window");
// Añadir un botón que elimine los elementos de menú.
var rem_listener = new Object();
rem_listener.click = function() {
  myMenuBar.removeAll();
};
remBtn.addEventListener("click", rem_listener);
```

MenuBar.removeMenuAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`menuBarInstance.removeMenuAt(`*`index`*`)`

Parámetros

`index` Índice del menú que se eliminará de la barra de menús.

Valor devuelto

Una referencia al menú en el índice especificado de la barra de menús. Si no hay ningún menú en esta posición de la barra de menús, el valor es `undefined`.

Descripción

Método; elimina el menú en el índice especificado. Si no hay ningún elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno. Además, cuando se elimina más de un menú, la asignación de índices se desplaza en consecuencia según se van eliminando menús.

Ejemplo

En el siguiente ejemplo se crea un menú File y un menú Edit en la barra de menús. Después, se llama a `removeMenuAt()` para eliminar el menú de la posición 0, que es el menú File, y se deja el menú Edit.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia `my_mb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
*/

import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;
```

```
var file_menu:Menu = my_mb.addMenu("File");
file_menu.addMenuItem({label:"New", instanceName:"newInstance"});
file_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

var edit_menu:Menu = my_mb.addMenu("Edit");
edit_menu.addMenuItem({label:"Cut", instanceName:"cutInstance"});
edit_menu.addMenuItem({label:"Copy", instanceName:"copyInstance"});
edit_menu.addMenuItem({label:"Paste", instanceName:"pasteInstance"});

//Borrar menú "file".
my_mb.removeMenuAt(0);
```

MenuBar.setMenuEnabledAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`menuBarInstance.setMenuEnabledAt(`*`index`*`, `*`boolean`*`)`

Parámetros

`index` Índice del elemento de menú que se definirá en la instancia de MenuBar.

`boolean` Valor booleano que indica si el elemento de menú del índice especificado está activado (`true`) o no (`false`).

Valor devuelto

Ninguno.

Descripción

Método; activa el menú en el índice especificado. Si no hay ningún menú en dicho índice, la llamada a este método no tiene efecto alguno.

Ejemplo

En el siguiente ejemplo se añade un menú File a la barra de menús y se llama al método `setMenuEnabledAt` para activar o desactivar el menú, según esté o no seleccionada la casilla de verificación menuEnabled_ch.

Arrastre una instancia del componente MenuBar al escenario y asígnele el nombre de instancia **my_mb** en el inspector de propiedades. Arrastre un componente CheckBox al escenario y asígnele el nombre de instancia **menuEnabled_ch**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente MenuBar en el escenario (nombre de instancia: my_mb)
  - Componente CheckBox en el escenario (nombre de instancia:
  menuEnabled_ch)
*/

import mx.controls.CheckBox;
import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;
var menuEnabled_ch:CheckBox;

menuEnabled_ch.selected = true;
var my_menu:Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

var chListener:Object = new Object();
chListener.click = function(evt_obj:Object) {
 // Activar o desactivar menú "file".
my_mb.setMenuEnabledAt(0, evt_obj.target.selected);
}
menuEnabled_ch.addEventListener("click", chListener);
```

CAPÍTULO 32

Componente NumericStepper

32

El componente NumericStepper permite recorrer un conjunto ordenado de números. El componente consta de un número en un cuadro de texto que se muestra junto a pequeños botones de flecha arriba y flecha abajo. Cuando el usuario presiona los botones, el número aumenta o disminuye de forma gradual según la unidad especificada en el parámetro `stepSize`, hasta que el usuario suelta los botones o hasta que se alcanza el valor máximo o mínimo. El texto del cuadro de texto del componente NumericStepper también se puede editar.

El componente NumericStepper sólo gestiona datos numéricos. Además, para ver más de dos lugares numéricos (por ejemplo, los números 5246 o 1,34), es preciso cambiar el tamaño del componente durante la edición.

Un componente NumericStepper puede estar activado o desactivado en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Si está activado y la selección interna está definida en el cuadro de texto, se selecciona al presionar el tabulador hasta su posición o al hacer clic en él. Cuando una instancia de NumericStepper está seleccionada, es posible utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Flecha abajo	El valor cambia en una unidad.
Flecha izquierda	Desplaza el punto de inserción a la izquierda dentro del cuadro de texto.
Flecha derecha	Desplaza el punto de inserción a la derecha dentro del cuadro de texto.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.
Flecha arriba	El valor cambia en una unidad.

Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia de NumericStepper refleja la configuración del parámetro value en el inspector de propiedades o el inspector de componentes durante la edición. Sin embargo, en la previsualización dinámica ni el teclado ni el ratón pueden interactuar con los botones de flecha de NumericStepper.

Cuando se añade el componente NumericStepper a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.NumericStepperAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Utilización del componente NumericStepper

Puede utilizar NumericStepper siempre que desee que el usuario seleccione un valor numérico. Por ejemplo, puede utilizar un componente NumericStepper en un formulario para que el usuario indique la fecha de caducidad de la tarjeta de crédito. Asimismo, puede utilizar un componente NumericStepper para que el usuario aumente o disminuya el tamaño de la fuente.

Parámetros de NumericStepper

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente NumericStepper en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

maximum define el valor máximo que se puede mostrar en el componente. El valor predeterminado es 10. Si establece un parámetro stepSize de forma que el valor mínimo más el valor de stepSize en algún punto no sea igual al valor máximo (mínimo + stepSize + stepSize + stepSize y así sucesivamente), el valor máximo *se mostrará* cuando el control numérico supere el valor máximo.

minimum define el valor mínimo que se puede mostrar en el componente. El valor predeterminado es 0.

stepSize define la unidad de incremento o disminución del componente con cada clic. El valor predeterminado es 1.

value define el valor que se muestra en el área de texto del componente. El valor predeterminado es 0.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente NumericStepper en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente NumericStepper utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase NumericStepper" en la página 1012.

Creación de aplicaciones con el componente NumericStepper

El siguiente procedimiento explica cómo añadir un componente NumericStepper a una aplicación durante la edición. El ejemplo coloca un componente NumericStepper y un componente Label en el escenario y crea un detector para un evento change en la instancia de NumericStepper. Cuando cambia el valor del control numérico, el ejemplo muestra el nuevo valor en la instancia de Label.

Para crear una aplicación con el componente NumericStepper:

1. Arrastre un componente NumericStepper desde el panel Componentes al escenario.
2. En el inspector de propiedades, introduzca el nombre de instancia **my_nstep**.
3. Arrastre un componente Label desde el panel Componentes al escenario.
4. En el inspector de propiedades, introduzca el nombre de instancia **my_label**.
5. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
/**
 Se requiere:
```

```
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
  - Componente Label en el escenario (nombre de instancia: my_label)
*/

var my_nstep:mx.controls.NumericStepper;
var my_label:mx.controls.Label;

my_label.text = "value = " + my_nstep.value;

//Crear un objeto detector.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 my_label.text = "value = " + evt_obj.target.value;
};

//Añadir detector.
my_nstep.addEventListener("change", nstepListener);
```

La última línea de código añade un controlador de eventos `change` a la instancia `my_nstep`. El controlador (`nstepListener`) asigna el valor actual del control numérico a la propiedad de text de la instancia de Label.

Personalización del componente NumericStepper

El componente NumericStepper puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o cualquier método o propiedad aplicable de la clase NumericStepper. (Véase "Clase NumericStepper" en la página 1012.)

Si se cambia el tamaño del componente NumericStepper, no cambia el tamaño de los botones de flecha arriba y flecha abajo. Si al componente NumericStepper se le asigna un tamaño superior a la altura predeterminada, los botones de flecha se fijan en la parte superior e inferior del componente. Los botones de flecha siempre aparecen a la derecha del cuadro de texto.

Utilización de estilos con el componente NumericStepper

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de NumericStepper. Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte "Utilización de estilos para personalizar el texto y el color de un componente" en *Utilización de componentes*.

Un componente NumericStepper admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"center"`.

Estilo	Tema	Descripción
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`repeatDelay`	Ambos	Número de milisegundos de demora entre el momento en que el usuario presiona por primera vez un botón y el momento en que comienza a repetirse la acción. El valor predeterminado es 500 (medio segundo).
`repeatInterval`	Ambos	Número de milisegundos entre clics automáticos cuando un usuario mantiene presionado el botón del ratón sobre un botón. El valor predeterminado es 35.
`symbolColor`	Sample	Color de las flechas. El valor predeterminado es 0x2B333C (gris oscuro).

Utilización de aspectos con el componente NumericStepper

El componente NumericStepper utiliza aspectos para representar sus estados de botón de flecha arriba y flecha abajo. Para aplicar aspectos al componente NumericStepper durante la edición, modifique los símbolos de aspecto en la carpeta Flash UI Components 2/Themes/MMDefault/Stepper Assets/States de la biblioteca. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

Si un componente NumericStepper está activado, los botones de flecha arriba y flecha abajo muestran el estado correspondiente al desplazamiento del puntero sobre ellos. Los botones muestran su estado presionado cuando se presionan. Cuando se suelta el ratón, recuperan el estado correspondiente al desplazamiento del puntero sobre ellos. Si el puntero se desplaza fuera de los botones estando el ratón presionado, los botones recuperan su estado original.

Si el componente está desactivado, muestra el estado desactivado sea cual sea la acción del usuario.

El componente NumericStepper admite las siguientes propiedades de aspecto:

Propiedad	Descripción
`upArrowUp`	Estado del botón de flecha arriba cuando no está presionado. El valor predeterminado es `StepUpArrowUp`.
`upArrowDown`	Estado del botón de flecha arriba cuando está presionado. El valor predeterminado es `StepUpArrowDown`.

Propiedad	Descripción
upArrowOver	Estado del botón de flecha arriba cuando se desplaza el puntero sobre él. El valor predeterminado es StepUpArrowOver.
upArrowDisabled	Estado del botón de flecha arriba cuando está desactivado. El valor predeterminado es StepUpArrowDisabled.
downArrowUp	Estado del botón de flecha abajo cuando no está presionado. El valor predeterminado es StepDownArrowUp.
downArrowDown	Estado del botón de flecha abajo cuando está presionado. El valor predeterminado es StepDownArrowDown.
downArrowOver	Estado del botón de flecha abajo cuando se desplaza el puntero sobre él. El valor predeterminado es StepDownArrowOver.
downArrowDisabled	Estado del botón de flecha abajo cuando está desactivado. El valor predeterminado es StepDownArrowDisabled.

Para crear símbolos de clip de película para aspectos de NumericStepper:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta Stepper Assets a la biblioteca del documento.
4. Expanda la carpeta Stepper Assets en la biblioteca del documento.
5. Expanda la carpeta Stepper Assets/States en la biblioteca del documento.
6. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo StepDownArrowDisabled.
7. Personalice el símbolo como desee.

   Por ejemplo, cambie el color blanco del gráfico interior por un gris claro.
8. Repita los pasos 6 y 7 en todos los símbolos que desee personalizar.

   Por ejemplo, repita el mismo cambio en la flecha arriba.
9. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
10. Arrastre un componente NumericStepper al escenario.

Este ejemplo ha personalizado los aspectos desactivados; utilice código ActionScript para establecer que la instancia de NumericStepper se desactive a fin de ver los aspectos modificados.

11. Seleccione Control > Probar película.

NOTA	La carpeta Stepper Assets/States también contiene un símbolo StepTrack, que se utiliza como espaciador entre los aspectos de flecha arriba y flecha abajo si la altura total de la instancia de NumericStepper es mayor que la suma de las dos alturas de flecha. Este identificador de vinculación de símbolos no se puede modificar mediante una propiedad de aspecto, aunque el símbolo de biblioteca puede modificarse si el identificador de vinculación permanece inalterado.

Clase NumericStepper

Herencia MovieClip > Clase UIObject > Clase UIComponent > NumericStepper

Nombre de clase de ActionScript mx.controls.NumericStepper

Las propiedades de la clase NumericStepper permiten definir lo siguiente en tiempo de ejecución: los valores mínimo y máximo que se muestran en el componente, la unidad de incremento o disminución como respuesta a un clic y el valor actual mostrado en el componente.

Si una propiedad de la clase NumericStepper se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente NumericStepper utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte “Creación de un desplazamiento personalizado de la selección” en *Utilización de componentes*.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.NumericStepper.version);
```

NOTA	El código `trace(myNumericStepperInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase NumericStepper

No hay métodos exclusivos de la clase NumericStepper.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase NumericStepper de la clase UIObject. Al llamar a estos métodos desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.methodName`.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase NumericStepper de la clase UIComponent. Al llamar a estos métodos desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.methodName`.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase NumericStepper

En la tabla siguiente se enumeran las propiedades de la clase NumericStepper.

Propiedad	Descripción
`NumericStepper.maximum`	Número que indica el valor máximo del rango.
`NumericStepper.minimum`	Número que indica el valor mínimo del rango.
`NumericStepper.nextValue`	Número que indica el siguiente valor de la secuencia. Es una propiedad de sólo lectura.
`NumericStepper.previousValue`	Número que indica el valor anterior de la secuencia. Es una propiedad de sólo lectura.
`NumericStepper.stepSize`	Número que indica la unidad de cambio de cada clic.
`NumericStepper.value`	Número que indica el valor actual del componente.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase NumericStepper de la clase UIObject. Al llamar a estas propiedades desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.`*`propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.

Propiedad	Descripción
UIObject.visible	Valor booleano que indica si el objeto es visible (true) o no (false).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase NumericStepper de la clase UIComponent. Al llamar a estas propiedades desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.propertyName`.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase NumericStepper

En la tabla siguiente se muestra el evento de la clase NumericStepper.

Evento	Descripción
NumericStepper.change	Se activa cuando cambia el valor del componente.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase NumericStepper de la clase UIObject. Al llamar a estos eventos desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.eventName`.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.

Evento	Descripción
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase NumericStepper de la clase UIComponent. Al llamar a estos eventos desde el objeto NumericStepper, debe utilizarse la forma `NumericStepper.eventName`.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

NumericStepper.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  //...
};
numericStepperInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se cambia el valor del componente.

Una instancia de componente (*stepperInstance*) distribuye un evento (en este caso, `change`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se crea un detector para un evento `change` en el control numérico llamado `my_nstep`. Cuando se cambia el valor del control numérico, el detector muestra el valor (propiedad `value`) en el panel Salida.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

// Crear un objeto detector.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object){
 // evt_obj.target es el componente que genera el evento change
 // es decir, el componente NumericStepper.
 trace("Value changed to " + evt_obj.target.value);
}
// Añadir detector.
my_nstep.addEventListener("change", nstepListener);
```

NumericStepper.maximum

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.maximum

Descripción

Propiedad; valor máximo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 10.

Ejemplo

En el ejemplo siguiente se define el valor máximo del rango del componente en 20.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/
var my_nstep:mx.controls.NumericStepper;

my_nstep.maximum = 20;
```

Véase también

NumericStepper.minimum

NumericStepper.minimum

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.minimum

Descripción

Propiedad; valor mínimo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se establece el valor mínimo y el valor inicial de la instancia de NumericStepper en 100 y el valor máximo en 120.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 100;
my_nstep.maximum = 120;
my_nstep.value = my_nstep.minimum;
```

Véase también

NumericStepper.maximum

NumericStepper.nextValue

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.nextValue

Descripción

Propiedad (sólo lectura); siguiente valor de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el siguiente ejemplo se establece el valor inicial de la instancia del componente NumericStepper en -6 y se establece la propiedad `stepSize` en 3. A continuación, se muestra el valor de la propiedad `nextValue` en el panel Salida. Debería verse el mismo valor al hacer clic en la flecha arriba del control numérico.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/
var my_nstep:mx.controls.NumericStepper;

my_nstep.stepSize = 3;
my_nstep.minimum = -6;
my_nstep.maximum = 12;
my_nstep.value = my_nstep.minimum;
trace(my_nstep.nextValue); // -3
```

Véase también

`NumericStepper.previousValue`

NumericStepper.previousValue

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.previousValue

Descripción

Propiedad (sólo lectura); valor anterior de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el siguiente ejemplo se establece el valor inicial de la instancia de NumericStepper en el valor mínimo de 6. Se define el valor `stepSize` en 3 y se crea un objeto detector para un evento change. Cuando se produce un evento change, el ejemplo muestra la propiedad `previousValue` en el panel Salida.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 6;
my_nstep.value = my_nstep.minimum;
my_nstep.maximum = 120;
my_nstep.stepSize = 3;

// Crear un objeto detector.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 trace("previous value = " + evt_obj.target.previousValue);
}

//Añadir detector.
my_nstep.addEventListener("change", nstepListener);
```

Véase también

NumericStepper.nextValue

NumericStepper.stepSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.stepSize

Descripción

Propiedad; cantidad en unidades que cambia con respecto al valor actual. El valor predeterminado es 1. Este valor no puede ser 0. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el siguiente ejemplo se establece el valor inicial de la instancia de NumericStepper en el valor mínimo de 3. También se define el valor `stepSize` en 3 para hacer que el control numérico produzca un incremento de 3 cuando el usuario haga clic en la flecha arriba y un decremento de 3 cuando el usuario haga clic en la flecha abajo.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 3;
my_nstep.maximum = 120;
my_nstep.value = my_nstep.minimum;
my_nstep.stepSize = 3;
```

NumericStepper.value

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

numericStepperInstance.value

Descripción

Propiedad; valor actual que se muestra en el área de texto del componente. El valor no se asigna si no corresponde al rango y al incremento de paso del componente definidos en la propiedad `stepSize`. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el ejemplo siguiente se define el valor actual de la instancia de NumericStepper en 10 y se envía el valor al panel Salida.

Arrastre una instancia del componente NumericStepper al escenario y asígnele el nombre de instancia `my_nstep` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente NumericStepper en el escenario (nombre de instancia:
  my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.value = 10;
my_nstep.maximum = 100;
trace(my_nstep.value); // 10
```

CAPÍTULO 33

Clase PopUpManager

33

Nombre de clase de ActionScript mx.managers.PopUpManager

La clase PopUpManager permite crear ventanas superpuestas que pueden ser modales o amodales (una ventana modal no permite interactuar con otras ventanas mientras está activa). Los métodos de esta clase se utilizan para crear y destruir ventanas emergentes.

Resumen de métodos de la clase PopUpManager

En la tabla siguiente se enumeran los métodos de la clase PopUpManager.

Método	Descripción
`PopUpManager.createPopUp()`	Crea una ventana emergente.
`PopUpManager.deletePopUp()`	Elimina una ventana emergente creada llamando a `PopUpManager.createPopUp()`.

PopUpManager.createPopUp()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
PopUpManager.createPopUp(parent, class, modal [, initobj, outsideEvents])
```

Parámetros

parent Referencia para que aparezca una ventana.

class Referencia a la clase de objeto que desea crear.

modal Valor booleano que indica si la ventana es modal (`true`) o no (`false`).

initobj Objeto que contiene propiedades de inicialización. Este parámetro es opcional.

outsideEvents Valor booleano que indica si se activa un evento cuando el usuario hace clic fuera de la ventana (`true`) o no (`false`). Este parámetro es opcional.

Valor devuelto

Referencia al objeto creado.

Si el parámetro *class* es Window y hay un componente Window en la biblioteca, la referencia devuelta es Window.

Descripción

Método; si es modal, una llamada a `createPopUp()` busca la ventana de nivel superior a partir de la principal y crea una instancia de clase. Si es amodal, una llamada a `createPopUp()` crea una instancia de clase secundaria con respecto a la ventana principal.

Ejemplo

El código siguiente crea una ventana modal cuando se hace clic en el botón:

```
lo = new Object();
lo.click = function(){
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
}
button.addEventListener("click", lo);
```

PopUpManager.deletePopUp()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

windowInstance`.deletePopUp();`

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina una ventana emergente y quita el estado modal. La ventana superpuesta es la encargada de llamar a `PopUpManager.deletePopUp()` cuando se está eliminando la ventana.

Ejemplo

El código siguiente crea una ventana modal denominada `win` con un botón de cierre, y elimina la ventana cuando se hace clic en el botón de cierre:

```
import mx.managers.PopUpManager
import mx.containers.Window
win = PopUpManager.createPopUp(_root, Window, true, {closeButton:true});
lo = new Object();
lo.click = function(){
  win.deletePopUp();
}
win.addEventListener("click", lo);
```

CAPÍTULO 34

Componente ProgressBar

34

El componente ProgressBar muestra el progreso de carga del contenido que se está cargando. El componente ProgressBar es útil para mostrar el estado de las imágenes y partes que se cargan de una aplicación. El proceso de carga puede ser determinado o indeterminado. Una barra de progreso *determinada* es la representación lineal del progreso de una tarea en el tiempo y se utiliza cuando la cantidad de contenido que va a cargarse es conocida. Una barra de progreso *indeterminada* se utiliza cuando la cantidad de contenido que va a cargarse es desconocida. Puede añadirse una etiqueta que muestre el progreso de carga del contenido que se está cargando.

El componente ProgressBar contiene un extremo izquierdo, un extremo derecho y una guía de progreso. Los extremos son simplemente los extremos de la barra de progreso, donde termina visualmente la barra de progreso. La previsualización dinámica de cada instancia de ProgressBar refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes. En la previsualización dinámica se reflejan los siguientes parámetros: conversion, direction, label, labelPlacement, mode y source.

Utilización del componente ProgressBar

Una barra de progreso permite ver el progreso de carga de un contenido. Esta información es fundamental para los usuarios mientras interactúan con una aplicación.

Hay varios modos de utilizar el componente ProgressBar; el modo se define con el parámetro mode. Los valores más frecuentes de este parámetro son event y polled. Estos modos utilizan el parámetro source para especificar un proceso de carga que emita eventos `progress` y `complete` (modo event y polled) o que revele los métodos `getBytesLoaded()` y `getsBytesTotal()` (modo polled). También puede utilizar el componente ProgressBar en modo manual si define manualmente las propiedades `maximum`, `minimum` e `indeterminate` junto con las llamadas al método `ProgressBar.setProgress()`.

Parámetros de ProgressBar

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ProgressBar en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

conversion es un número por el que se dividen los valores %1 y %2 de la cadena de la etiqueta antes de que se muestren. El valor predeterminado es 1.

direction indica el sentido en el que se rellena la barra de progreso. Este valor puede ser `right` o `left`; el valor predeterminado es `right`.

label es el texto que indica el progreso de la carga. Este parámetro es una cadena con el formato "%1 de %2 cargado (%3%%)". En esta cadena, %1 es un marcador de posición de los bytes actuales cargados, %2 un marcador de posición de los bytes totales cargados y %3 un marcador de posición del porcentaje del contenido cargado. Los caracteres "%%" son un marcador de posición del carácter "%". Si el valor de %2 es desconocido, se sustituye por dos signos de interrogación (??). Si el valor es undefined, la etiqueta no se muestra.

labelPlacement indica la posición de la etiqueta con respecto a la barra de progreso. Este parámetro puede tener uno de estos valores: `top`, `bottom`, `left`, `right` y `center`. El valor predeterminado es `bottom`.

mode es el modo en el que funciona la barra de progreso. Este valor puede ser uno de los siguientes: `event`, `polled` o `manual`. El valor predeterminado es `event`.

source es una cadena que se convertirá en el objeto que represente el nombre de instancia del origen.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente ProgressBar en el inspector de componentes (a través de la opción de menú Ventana > Inspector de componentes):

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente ProgressBar utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase ProgressBar" en la página 1037.

Creación de aplicaciones con el componente ProgressBar

En el procedimiento siguiente se explica cómo añadir un componente ProgressBar a una aplicación durante la edición. En este ejemplo, la barra de progreso se utiliza en modo event. En el modo event, el contenido que se carga debe emitir los eventos `progress` y `complete` que la barra de progreso utiliza para mostrar el progreso. El componente Loader emite estos eventos; para más información, consulte "Componente Loader" en la página 841.

Para crear una aplicación con el componente ProgressBar en modo event:

1. Arrastre un componente ProgressBar desde el panel Componentes al escenario.
2. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **my_pb**.
   - Seleccione Event para el parámetro mode.
3. Arrastre un componente Loader desde el panel Componentes al escenario.
4. En el inspector de propiedades, introduzca el nombre de instancia **my_ldr**.
5. En el escenario, seleccione la barra de progreso; en el inspector de propiedades, introduzca **my_ldr** para el parámetro source.
6. Seleccione el fotograma 1 en la línea de tiempo, abra el panel Acciones e introduzca el código siguiente para cargar un archivo JPEG en el componente Loader:

```
/**
 Se requiere:
  - Componente Loader en el escenario (nombre de instancia: my_ldr)
  - Componente ProgressBar en el escenario (nombre de instancia: my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.source = my_ldr;
my_ldr.autoLoad = false;
my_ldr.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// cuando autoLoad es false, la carga no comienza hasta haber invocado el
  método load()
my_ldr.load();
```

En el ejemplo siguiente se utiliza la barra de progreso en modo polled. En este modo, ProgressBar utiliza los métodos `getBytesLoaded()` y `getBytesTotal()` del objeto de origen para mostrar el progreso de la operación.

Para crear una aplicación con el componente ProgressBar en modo polled:

1. Arrastre un componente ProgressBar desde el panel Componentes al escenario.
2. En el inspector de propiedades, introduzca el nombre de instancia **my_pb**.
3. Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente para crear un objeto Sound denominado `my_sound` y llamar al método `loadSound()` a fin de cargar un sonido en el objeto Sound:

```
/**
 Se requiere:
  - Componente ProgressBar en el escenario (nombre de instancia: my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = "my_sound";

var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 trace("Sound loaded");
}
my_pb.addEventListener("complete", pbListener);

var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3",
  true);
```

En el ejemplo siguiente se utiliza la barra de progreso en modo manual. Para mostrar el progreso de la operación al utilizar el modo manual, debe definir las propiedades `maximum`, `minimum` e `indeterminate` junto con el método `setProgress()`, pero no necesita definir la propiedad `source`.

Para crear una aplicación con el componente ProgressBar en modo manual:

1. Arrastre un componente ProgressBar desde el panel Componentes al escenario.
2. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **my_pb**.
   - Seleccione Manual para el parámetro mode.

3. Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente para actualizar la barra de progreso de forma manual cada vez que se descarga un archivo con una llamada a `setProgress()`:

```
for (var i:Number = 1; i <= total; i++){
  // insertar código para cargar archivo
  my_pb.setProgress(i, total);
}
```

A continuación se muestran otros dos ejemplos.

Para crear una aplicación con el componente ProgressBar en modo manual (ejemplo 2):

1. Arrastre un componente Label al escenario y asígnele el nombre de instancia **my_label**.
2. Arrastre un componente ProgressBar al escenario y asígnele el nombre de instancia **my_pb**.
3. Seleccione la instancia my_pb de ProgressBar en el escenario y, en el inspector de propiedades, establezca el parámetro mode del componente en "manual".
4. Seleccione el fotograma 1 de la línea de tiempo y añada el código ActionScript siguiente en el panel Acciones:

```
var feed_xml:XML = new XML();
feed_xml.onLoad = function(success:Boolean):Void {
  clearInterval(timer);
  my_label.text = "XML Loaded";
  my_pb.setProgress(feed_xml.getBytesLoaded(),
  feed_xml.getBytesTotal());
};
function updatePB(local_xml:XML):Void {
  my_pb.setProgress(local_xml.getBytesLoaded(),
  local_xml.getBytesTotal());
}
var timer:Number = setInterval(updatePB, 100, feed_xml);
feed_xml.load("http://www.helpexamples.com/flash/xml/menu.xml");
```

5. Pulse Control+Intro para probar.

Para crear una aplicación con el componente ProgressBar en modo manual (ejemplo 3):

1. Arrastre un componente ProgressBar al escenario y asígnele el nombre de instancia **my_pb**.
2. Seleccione la instancia my_pb de ProgressBar en el escenario y, en el inspector de propiedades, establezca el parámetro mode del componente en "manual".

3. Seleccione el fotograma 1 de la línea de tiempo y añada el código ActionScript siguiente en el panel Acciones:

```
var img_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number) {
  my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
//my_pb._visible = false;
};
img_mcl.addListener(mclListener);
this.createEmptyMovieClip("image_mc", 20);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

NOTA	puede quitar la marca de comentario de la línea //my_pb._visible = false si desea ocultar el componente una vez que se haya cargado el contenido.

4. Pulse Control+Intro para probar.

Personalización del componente ProgressBar

El componente ProgressBar puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()`.

Los extremos izquierdo y derecho de la barra de progreso y el gráfico de la guía deslizamiento tienen un tamaño fijo. Cuando se cambia el tamaño de una barra de progreso, la parte central de ésta se ajusta a la distancia entre los dos extremos. Si la barra de progreso es demasiado pequeña, es posible que no se pueda representar correctamente.

Utilización de estilos con el componente ProgressBar

Se pueden definir propiedades de estilo para cambiar el aspecto de una instancia de la barra de progreso. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

Un componente ProgressBar admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`barColor`	Sample	Color en primer plano que indica el porcentaje completado. El color predeterminado es blanco. Para definir el color de la barra en un componente con el tema Halo, establezca la propiedad de estilo `themeColor`.
`trackColor`	Sample	Color de fondo de la barra. El valor predeterminado es 0x666666 (gris oscuro).

Utilización de aspectos con el componente ProgressBar

El componente ProgressBar utiliza aspectos para representar la guía de la barra de progreso, la barra completada, así como una barra indeterminada. Para aplicar aspectos al componente ProgressBar durante la edición, modifique los símbolos en la carpeta Flash UI Components 2/Themes/MMDefault/ProgressBar Elements. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

Los gráficos de la guía de deslizamiento y la barra se componen de tres aspectos que corresponden a los extremos izquierdo y derecho, y la parte central. Los extremos se utilizan "tal cual" y la parte central adapta su tamaño horizontal para ajustarse a la anchura de la instancia de ProgressBar.

La barra indeterminada se utiliza cuando la propiedad `indeterminate` de la instancia de ProgressBar se establece en `true`. El aspecto adapta su tamaño horizontal para ajustarse a la anchura de la barra de progreso.

El componente ProgressBar admite las siguientes propiedades de aspecto:

Propiedad	Descripción
`progTrackMiddleName`	Parte central ampliable de la guía de deslizamiento. El valor predeterminado es `ProgTrackMiddle`.
`progTrackLeftName`	Extremo izquierdo con tamaño fijo. El valor predeterminado es `ProgTrackLeft`.
`progTrackRightName`	Extremo derecho con tamaño fijo. El valor predeterminado es `ProgTrackRight`.
`progBarMiddleName`	Gráfico de barra central ampliable. El valor predeterminado es `ProgBarMiddle`.
`progBarLeftName`	Extremo izquierdo de la barra con tamaño fijo. El valor predeterminado es `ProgBarLeft`.
`progBarRightName`	Extremo derecho de la barra con tamaño fijo. El valor predeterminado es `ProgBarRight`.
`progIndBarName`	Gráfico de barra indeterminado. El valor predeterminado es `ProgIndBar`.

Para crear símbolos de clip de película para aspectos de ProgressBar:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta ProgressBar Assets a la biblioteca del documento.
4. Expanda la carpeta ProgressBar Assets/Elements en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo ProgIndBar.
6. Personalice el símbolo como desee.

   Por ejemplo, gire la guía de deslizamiento horizontalmente.
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Arrastre un componente ProgressBar al escenario.

   Si desea ver los aspectos modificados de este ejemplo, utilice código ActionScript para establecer la propiedad `indeterminate` en `true`.
10. Seleccione Control > Probar película.

Clase ProgressBar

Herencia MovieClip > Clase UIObject > ProgressBar

Nombre de clase de ActionScript mx.controls.ProgressBar

Si una propiedad de la clase ProgressBar se define con ActionScript, se sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.ProgressBar.version);
```

NOTA	El código `trace(myProgressBarInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase ProgressBar

En la tabla siguiente se muestra el método de la clase ProgressBar.

Método	Descripción
`ProgressBar.setProgress()`	Define el estado de la barra de progreso para que refleje la cantidad de progreso realizado cuando la barra está en modo manual.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase ProgressBar de la clase UIObject. Al llamar a estos métodos desde el objeto ProgressBar, debe utilizarse la forma `ProgressBar.`*`methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Resumen de propiedades de la clase ProgressBar

En la tabla siguiente se enumeran las propiedades de la clase ProgressBar.

Propiedad	Descripción
`ProgressBar.conversion`	Número utilizado para convertir el valor de los bytes actuales cargados y el valor total de los bytes cargados.
`ProgressBar.direction`	Dirección en que se rellena la barra de progreso.
`ProgressBar.indeterminate`	Indica si se desconoce el tamaño del origen que se está cargando.
`ProgressBar.label`	Texto que acompaña a la barra de progreso.
`ProgressBar.labelPlacement`	Ubicación de la etiqueta con respecto a la barra de progreso.
`ProgressBar.maximum`	Valor máximo de la barra de progreso en modo manual.
`ProgressBar.minimum`	Valor mínimo de la barra de progreso en modo manual.
`ProgressBar.mode`	Modo en el que la barra de progreso carga el contenido.
`ProgressBar.percentComplete`	Sólo lectura; número que indica el porcentaje cargado.
`ProgressBar.source`	Contenido de la carga.
`ProgressBar.value`	Sólo lectura; indica la cantidad del progreso realizado.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase ProgressBar de la clase UIObject. Al llamar a estas propiedades desde el objeto ProgressBar, debe utilizarse la forma `ProgressBar.`*`propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Resumen de eventos de la clase ProgressBar

En la tabla siguiente se enumeran los eventos de la clase ProgressBar.

Evento	Descripción
`ProgressBar.complete`	Se activa cuando se completa la carga.
`ProgressBar.progress`	Se activa a medida que se carga el contenido en modo manual o polled.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase ProgressBar de la clase UIObject. Al llamar a estos eventos desde el objeto ProgressBar, debe utilizarse la forma `ProgressBar.`*`eventName`*.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

ProgressBar.complete

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object) {
  // ...
};
progressBarInstance.addEventListener("complete", listenerObject);
```

Sintaxis 2:

```
on (complete) {
  // ...
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, existen dos propiedades adicionales definidas para el evento `ProgressBar.complete`: `current` (el valor cargado igual al total) y `total` (el valor total).

Descripción

Evento; se difunde a todos los detectores cuando se ha completado el progreso de carga.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*`progressBarInstance`*) distribuye un evento (en este caso, `complete`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de ProgressBar. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `progressBarInstance`, envía "_level0.progressBarInstance" al panel Salida:

```
on (complete) {
  trace(this);
}
```

Ejemplo

En este ejemplo se crea un componente Loader, un componente ProgressBar (`my_pb`) para él y un detector que vuelve la barra de progreso invisible cuando se produce el evento `complete`. El ejemplo carga una imagen en el cargador `my_ldr`.

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.Loader, "my_ldr", 10, {autoLoad:false});
this.createClassObject(mx.controls.ProgressBar, "my_pb", 20,
  {indeterminate:true, source:my_ldr, mode:"polled"});

// Crear un objeto detector
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 my_pb.visible = false;
};
// Añadir detector
my_pb.addEventListener("complete", pbListener);

my_ldr.load("http://www.helpexamples.com/flash/images/image2.jpg");
```

Véase también

`EventDispatcher.addEventListener()`

ProgressBar.conversion

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.conversion

Descripción

Propiedad; número que define un valor de conversión para los valores entrantes. Divide los valores actual y total, los redondea a la baja y muestra el valor convertido en la propiedad `label`. El valor predeterminado es 1.

NOTA	El valor redondeado a la baja es el valor entero más cercano inferior o igual al valor especificado. Por ejemplo, el número 4,6 se convierte en 4.

Ejemplo

El siguiente código muestra el progreso de carga de un objeto de sonido dividiendo el número de bytes cargados por un valor de conversión de 1024 para producir un valor en kilobytes.

Primero hay que arrastrar un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 20);

//Establecer atributos de barra de progreso
my_pb.mode = "polled";
my_pb.source = "my_sound";
my_pb.label = "%1 kb loaded";
my_pb.conversion = 1024;

//Cargar sonido
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3",
  true);
```

ProgressBar.direction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.direction

Descripción

Propiedad; indica la dirección de relleno de la barra de progreso. Un valor de `right` especifica que la barra se rellenará de izquierda a derecha. Un valor de `left` especifica que la barra se rellenará de derecha a izquierda. El valor predeterminado es `right`.

Ejemplo

El siguiente código carga un objeto de sonido y marca el progreso con una barra de progreso que se rellena hacia la izquierda.

Primero hay que arrastrar un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 20);

//Establecer atributos de barra de progreso
my_pb.mode = "polled";
my_pb.source = "my_sound";
my_pb.direction = "left";

//Cargar sonido
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3",
  true);
```

ProgressBar.indeterminate

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.indeterminate

Descripción

Propiedad; valor booleano que indica si la barra de progreso tiene un relleno discontinuo y un origen de carga de tamaño desconocido (`true`), o un relleno continuo y un origen de carga con un tamaño conocido (`false`). Por ejemplo, puede evitar el uso de esta propiedad si está cargando un conjunto de datos grande en un archivo SWF y desconoce el tamaño de los datos de la carga.

Ejemplo

El código siguiente crea una barra de progreso indeterminada que se mueve de izquierda a derecha con un relleno discontinuo:

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

//Crear un objeto detector
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
trace("Height: " + evt_obj.target.height + ", Width: " +
  evt_obj.target.width);};
//Añadir detector
my_pb.addEventListener("complete", pbListener);
```

```
//Establecer configuración de barra de progreso
my_pb.mode = "polled";
my_pb.indeterminate = true;
my_pb.source = my_ldr;

//Establecer configuración de cargador
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.move(100, 100)
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.label

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.label

Descripción

Propiedad; texto que indica el progreso de carga. Este parámetro es una cadena con el formato `"%1 de %2 cargado (%3%%)"`. En esta cadena, `%1` es un marcador de posición de los bytes actuales cargados, `%2` un marcador de posición de los bytes totales cargados y `%3` un marcador de posición del porcentaje del contenido cargado. Los caracteres `%%` permiten que Flash muestre un solo carácter `%`. Si el valor de `%2` es desconocido, se sustituye por `??`. Si el valor es `undefined`, la etiqueta no se muestra. El valor predeterminado es `"LOADING %3%%"`.

Ejemplo

En el siguiente ejemplo se carga una imagen en un cargador y se marca el progreso con una barra de progreso cuya etiqueta especifica el porcentaje del total de kilobytes que se han cargado.

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/
```

```
System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

my_ldr.move(0, 30);

//Establecer configuración de barra de progreso
my_pb.mode = "polled";
my_pb.source = my_ldr;
my_pb.label = "%1 of %2 KB loaded";
my_pb.conversion = 1024; // 1024 bytes en un KB

//Establecer configuración de cargador
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg")
```

Véase también

`ProgressBar.labelPlacement`

ProgressBar.labelPlacement

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`progressBarInstance``.labelPlacement`

Descripción

Propiedad; define la ubicación de la etiqueta con respecto a la barra de progreso. Los valores posibles son `"left"`, `"right"`, `"top"`, `"bottom"` y `"center"`.

Ejemplo

En el siguiente ejemplo se carga una imagen en un cargador y se marca el progreso con una barra de progreso. Se define la propiedad `labelPlacement` en `top` para colocar la etiqueta por encima de la barra de progreso.

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

my_ldr.move(0, 30);

// Establecer configuración de barra de progreso
my_pb.mode = "polled";
my_pb.source = my_ldr;
my_pb.label = "%1 of %2 KB loaded";
my_pb.conversion = 1024; // 1024 bytes en un KB
my_pb.labelPlacement = "top";

// Establecer configuración de cargador
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Véase también

`ProgressBar.label`

ProgressBar.maximum

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.maximum

Descripción

Propiedad; valor más alto para la barra de progreso cuando la propiedad `ProgressBar.mode` está definida en `"manual"`.

Ejemplo

En el siguiente ejemplo se incrementa un componente ProgressBar de forma manual hasta un valor `maximum` de 200, número en el que se detiene. Se muestra el incremento en el panel Salida mientras va aumentando el valor.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia `my_pb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente ProgressBar en el escenario (nombre de instancia: my_pb)
*/

var my_pb:mx.controls.ProgressBar;

//Establecer modo de barra de progreso
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

//valor numérico mínimo antes de que incremente la barra de progreso
my_pb.minimum = 100;

//valor máximo de barra de progreso antes de que se detenga
my_pb.maximum = 200;

var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  //actualizar progreso de número en incremento
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

Véase también

`ProgressBar.minimum`, `ProgressBar.mode`

ProgressBar.minimum

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.minimum

Descripción

Propiedad; valor más bajo para la barra de progreso cuando la propiedad `ProgressBar.mode` está definida en `"manual"`.

Ejemplo

En el siguiente ejemplo se incrementa de forma manual un componente ProgressBar, comenzando con un valor mínimo de 100. Se muestra el incremento en el panel Salida mientras el valor va aumentando.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia `my_pb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Componente ProgressBar en el escenario (nombre de instancia: my_pb)
*/

var my_pb:mx.controls.ProgressBar;

//Establecer modo de barra de progreso
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

//valor numérico mínimo antes de que incremente la barra de progreso
my_pb.minimum = 100;

//valor máximo de barra de progreso antes de que se detenga
my_pb.maximum = 200;
```

```
var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  //actualizar progreso de número en incremento
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

Véase también

`ProgressBar.maximum`, `ProgressBar.mode`

ProgressBar.mode

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`progressBarInstance``.mode`

Descripción

Propiedad; modo en el que la barra de progreso carga el contenido. Este valor puede ser `"event"`, `"polled"` o `"manual"`.

Los modos más frecuentes son event y polled. En el modo event, la propiedad `source` especifica el contenido de carga que emite eventos `progress` y `complete`; debe utilizar un objeto Loader en este modo. En el modo polled, la propiedad `source` especifica el contenido de carga (como un objeto MovieClip) que muestra métodos `getBytesLoaded()` y `getsBytesTotal()`. Todo objeto que muestre estos métodos puede utilizarse como origen en modo polled (incluido un objeto personalizado o la línea de tiempo raíz).

También puede utilizar el componente ProgressBar en modo manual si define manualmente las propiedades `maximum`, `minimum` e `indeterminate`, y llama al método `ProgressBar.setProgress()`.

Ejemplo

En el siguiente ejemplo se carga una imagen en un cargador y se marca el progreso de carga con una barra de progreso que se establece en el modo `event`. Cuando finaliza la carga, un detector del evento `complete` muestra el nombre del objeto cargador.

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

//Crear un objeto detector
var ldrListener:Object = new Object();
ldrListener.complete = function(evt_obj:Object) {
  trace("Event complete for " + evt_obj.target);
};
//Añadir detector
my_ldr.addEventListener("complete", ldrListener);

//Establecer configuración de barra de progreso
my_pb.mode = "event";
my_pb.indeterminate = true;
my_pb.source = my_ldr;

//Establecer configuración de cargador
my_ldr.move(0,30);
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.percentComplete

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.percentComplete

Descripción

Propiedad (sólo lectura); indica el porcentaje del contenido que se ha cargado. Este valor se redondea a la baja. El valor redondeado a la baja es el valor entero más cercano inferior o igual al valor especificado. Por ejemplo, el número 7,8 se convierte en 7. La fórmula siguiente se utiliza para calcular el porcentaje:

```
100 * (value - minimum) / (maximum - minimum)
```

Ejemplo

En el siguiente ejemplo se carga una imagen en un cargador que está asociado con una barra de progreso. Un detector para el evento `progress` y otro para el evento `complete` acceden a la propiedad `percentComplete` para mostrar el porcentaje de carga que se ha completado.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia `my_pb` en el inspector de propiedades. Arrastre una instancia del componente Loader al escenario y asígnele el nombre de instancia `my_ldr` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Instancia del componente Loader en el escenario (nombre de instancia:
  my_ldr)
  - Instancia del componente Progress en el escenario (nombre de instancia:
  my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;

var pbListener:Object = new Object();
pbListener.progress = function(evt_obj:Object) {
 trace("progress = " + my_pb.percentComplete + "%");
}
pbListener.complete = function(evt_obj:Object) {
 trace("complete = " + my_pb.percentComplete + "%");
}
my_pb.addEventListener("progress", pbListener);
my_pb.addEventListener("complete", pbListener);

// cuando autoLoad es false, la carga no comienza hasta haber invocado el
  método load()
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.progress

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object) {
  // ...
};
progressBarInstance.addEventListener("progress", listenerObject);
```

Sintaxis 2:

```
on (progress) {
  // ...
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, existen dos propiedades adicionales definidas para el evento `ProgressBar.progress`: `current` (el valor cargado igual a `total`) y `total` (el valor total).

Descripción

Evento; se difunde a todos los detectores registrados siempre que cambia el valor de una barra de progreso.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*`progressBarInstance`*) distribuye un evento (en este caso, `progress`) y éste se controla mediante una función, también denominada *controlador*, en un objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de ProgressBar. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `progressBarInstance`, envía "_level0.`progressBarInstance`" al panel Salida:

```
on (progress) {
  trace(this);
}
```

Ejemplo

En este ejemplo se carga una imagen en un cargador con una barra de progreso asociada y se crea un detector para el evento `progress`. Cuando se produce el evento progress, el ejemplo muestra la propiedad `value`, que es un valor entre `ProgressBar.minimum` y `ProgressBar.maximum`.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia `my_pb` en el inspector de propiedades. Arrastre una instancia del componente Loader al escenario y asígnele el nombre de instancia `my_ldr` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Instancia del componente Loader en el escenario (nombre de instancia:
  my_ldr)
  - Instancia del componente Progress en el escenario (nombre de instancia:
  my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;

//Crear un objeto detector
var pbListener:Object = new Object();
pbListener.progress = function(evt_obj:Object) {
  // evt_obj.target es el componente que genera el evento progress,
  // es decir, la barra de progreso.
  trace("Current progress value = " + evt_obj.target.value);
};
//Añadir detector
my_pb.addEventListener("progress", pbListener);

// cuando autoLoad es false, la carga no comienza hasta haber invocado el
  método load()
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Véase también

`EventDispatcher.addEventListener()`

ProgressBar.setProgress()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`progressBarInstance``.setProgress(`*`completed, total`*`)`

Parámetros

`completed` Número que indica la cantidad del progreso realizado. Se pueden utilizar las propiedades `ProgressBar.label` y `ProgressBar.conversion` para mostrar el número en valor porcentual o cualquier unidad que seleccione, en función del origen de la barra de progreso.

`total` Número que indica el progreso que debe realizarse hasta alcanzar el 100%.

Valor devuelto

Número que indica la cantidad del progreso realizado.

Descripción

Método; define el estado de la barra de progreso para que refleje la cantidad de progreso realizado cuando la propiedad `ProgressBar.mode` está definida en `"manual"`. Puede hacer una llamada a este método para que la barra refleje el estado de un proceso que no sea de carga. Por ejemplo, puede definir explícitamente la barra de progreso con un progreso cero.

El parámetro *`completed`* se asigna a la propiedad `value` y el parámetro *`total`* se asigna a la propiedad `maximum`. La propiedad `minimum` no cambia.

Ejemplo

En el siguiente ejemplo se establece el modo de barra de progreso en `manual` y se llama a `setProgress()` desde la función `onEnterFrame()`, que se invoca repetidamente en la velocidad de fotogramas del archivo SWF. El ejemplo establece el valor mínimo para la barra de progreso en 100 y el máximo en 200 y marca el progreso en incrementos de 1.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia `my_pb` en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - ProgressBar en el escenario (nombre de instancia: my_pb)
*/

var my_pb:mx.controls.ProgressBar;

//Establecer modo de barra de progreso
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

//valor numérico mínimo antes de que incremente la barra de progreso
my_pb.minimum = 100;

//valor máximo de barra de progreso antes de que se detenga
my_pb.maximum = 200;

var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  //actualizar progreso de número en incremento
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

ProgressBar.source

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.source

Descripción

Propiedad; referencia a la instancia que se va a cargar y de la que mostrará el proceso de carga. El contenido que se carga debe emitir un evento `progress` del que se obtienen los valores total y actual. Este evento sólo se difunde cuando `ProgressBar.mode` está definida en `"event"` o `"polled"`. El valor predeterminado es`undefined`.

El componente ProgressBar se puede utilizar con el contenido de una aplicación, incluido `_root`.

Ejemplo

En el siguiente ejemplo se carga una imagen en un cargador y se marca el progreso con una barra de progreso. El ejemplo establece la propiedad `source` en el nombre del componente Loader (`my_ldr`) para asociar el contenido con la barra de progreso.

Primero hay que arrastrar un componente Loader y un componente ProgressBar desde el panel Componentes a la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente ProgressBar en la biblioteca
  - Componente Loader en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

//Crear un objeto detector
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 evt_obj.target.visible = false;
};
//Añadir detector
my_pb.addEventListener("complete", pbListener);

//Establecer configuración de barra de progreso
my_pb.mode = "polled";
my_pb.indeterminate = true;
my_pb.source = my_ldr;

//Establecer configuración de cargador
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Véase también

`ProgressBar.mode`

ProgressBar.value

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

progressBarInstance.value

Descripción

Propiedad (sólo lectura); indica la cantidad del progreso realizado. La propiedad es un número entre el valor de ProgressBar.minimum y ProgressBar.maximum. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se carga una imagen en un componente Loader y se marca el progreso con una barra de progreso. Cuando finaliza la carga, el ejemplo muestra los valores mínimo, máximo y actual de la barra de progreso.

Arrastre una instancia del componente ProgressBar al escenario y asígnele el nombre de instancia my_pb en el inspector de propiedades. Arrastre una instancia del componente Loader al escenario y asígnele el nombre de instancia my_ldr en el inspector de propiedades. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
/**
 Se requiere:
  - Instancia del componente Loader en el escenario (nombre de instancia:
  my_ldr)
  - Instancia del componente Progress en el escenario (nombre de instancia:
  my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;
```

```
//Crear un objeto detector
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object){
  // event_obj.target es el componente que genera el evento complete,
  // es decir, la barra de progreso.
  trace("Minimum value is: " + evt_obj.target.minimum + " bytes");
  trace("Maximum value is: " + evt_obj.target.maximum + " bytes");
  trace("Current ProgressBar value = " + evt_obj.target.value + " bytes");
}
//Añadir detector
my_pb.addEventListener("complete", pbListener);

// cuando autoLoad es false, la carga no comienza hasta haber invocado el
  método load()
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

CAPÍTULO 35

35 Componente RadioButton

El componente RadioButton permite obligar al usuario a seleccionar una opción de un conjunto de opciones. Este componente debe utilizarse en un grupo formado al menos por dos instancias de RadioButton. Sólo es posible seleccionar un miembro del grupo cada vez. Al seleccionar un botón de opción de un grupo, se anula la selección del botón de opción seleccionado en dicho grupo. Puede definir el parámetro `groupName` para indicar el grupo al que pertenece el botón de opción.

Los botones de opción pueden estar activados o desactivados. Un botón de opción en estado desactivado no recibe la entrada del ratón ni del teclado. Si el usuario hace clic en un grupo de componentes RadioButton o presiona el tabulador hasta su posición, la selección recae sólo en el botón de opción elegido. A continuación, el usuario puede utilizar las siguientes teclas para controlarlo:

Tecla	Descripción
Flecha arriba/ flecha izquierda	La selección se desplaza al botón anterior del grupo de botones de opción.
Flecha abajo/ flecha derecha	La selección se desplaza al botón siguiente del grupo de botones de opción.
Tabulador	Desplaza la selección desde el grupo de botones de opción hasta el componente siguiente.

Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia de RadioButton del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el inspector de componentes. Sin embargo, la previsualización dinámica no muestra la exclusión mutua de la selección. Si define el parámetro seleccionado en true para dos botones de opción del mismo grupo, ambos botones aparecerán seleccionados, si bien sólo la última instancia creada estará seleccionada en tiempo de ejecución. Para más información, consulte "Parámetros de RadioButton" en la página 1062.

Cuando se añade el componente RadioButton a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.RadioButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, "Creación de contenido accesible" en *Utilización de Flash*.

Utilización del componente RadioButton

Los botones de opción son una parte fundamental de cualquier formulario o aplicación Web. Puede utilizar los botones de opción siempre que necesite que un usuario seleccione un elemento de un grupo de opciones. Por ejemplo, podría emplear botones de opción en un formulario para preguntar qué tarjeta de crédito desea utilizar el cliente.

Parámetros de RadioButton

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente RadioButton en el inspector de propiedades o en el inspector de componentes:

data es el valor de los datos asociados con el botón de opción. No hay un valor predeterminado.

groupName es el nombre del grupo del botón de opción. El valor predeterminado es `radioGroup`.

label establece el valor del texto en el botón. El valor predeterminado es `Radio Button`.

labelPlacement orienta el texto de la etiqueta que muestra el botón. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom` El valor predeterminado es `right`. Para más información, consulte `RadioButton.labelPlacement`.

selected define el valor inicial del botón de opción en seleccionado (`true`) o no (`false`). Un botón de opción seleccionado muestra un punto en su interior. Sólo un botón de opción de un grupo puede tener un valor de selected definido en `true`. Si el grupo contiene más de un botón de opción definido en `true`, sólo se seleccionará el botón cuya instancia se haya creado en último lugar. El valor predeterminado es `false`.

Puede escribir código ActionScript para definir otras opciones para las instancias de RadioButton con los métodos, propiedades y eventos de la clase RadioButton. Para más información, consulte "Clase RadioButton" en la página 1068.

Creación de aplicaciones con el componente RadioButton

El procedimiento siguiente explica cómo añadir componentes RadioButton a una aplicación durante la edición. En este ejemplo, los botones de opción se utilizan para presentar una pregunta "¿Es aficionado a Flash?", que requiere una respuesta afirmativa o negativa. Los datos del grupo de botones de opción aparecen en un componente TextArea con el nombre de instancia `theVerdict`.

Para crear una aplicación con el componente RadioButton:

1. Arrastre dos componentes RadioButton desde el panel Componentes al escenario.
2. Seleccione uno de los botones de opción. En el inspector de componentes, siga este procedimiento:
   - Introduzca **Sí** para el parámetro de etiqueta.
   - Introduzca **Aficionado a Flash** para el parámetro de datos.
3. Seleccione el otro botón de opción. En el inspector de componentes, siga este procedimiento:
   - Introduzca **No** para el parámetro de etiqueta.
   - Introduzca **Anti-Flash** para el parámetro de datos.
4. Arrastre un componente TextArea desde el panel Componentes al escenario y asígnele el nombre de instancia `theVerdict`.
5. Seleccione el fotograma 1 de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
flashistListener = new Object();
flashistListener.click = function (evt){
    theVerdict.text = evt.target.selection.data
}
radioGroup.addEventListener("click", flashistListener);
```

La última línea del código añade un controlador de eventos `click` al grupo de botones de opción `radioGroup`. El controlador define la propiedad `text` de la instancia `theVerdict` del componente TextArea con el valor de la propiedad `data` del botón de opción seleccionado en el grupo de botones de opción `radioGroup`. Para más información, consulte `RadioButton.click`.

Personalización del componente RadioButton

El componente RadioButton puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`).

El recuadro de delimitación de un componente RadioButton es invisible y designa el área activa del componente. Si aumenta el tamaño del componente, aumenta también el tamaño del área activa.

Si el tamaño de la etiqueta supera el del recuadro de delimitación del componente, la etiqueta se recorta para ajustarse al espacio disponible.

Utilización de estilos con el componente RadioButton

Puede definir propiedades de estilo para cambiar el aspecto de una instancia de RadioButton. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

El componente RadioButton utiliza los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`symbolBackgroundColor`	Sample	Color de fondo del botón de opción. El valor predeterminado es 0xFFFFFF (blanco).

Estilo	Tema	Descripción
symbolBackgroundDisabledColor	Sample	Color de fondo del botón de opción cuando está desactivado. El valor predeterminado es 0xEFEEEF (gris claro).
symbolBackgroundPressedColor	Sample	Color de fondo del botón de opción cuando se presiona. El valor predeterminado es 0xFFFFFF (blanco).
symbolColor	Sample	Color del punto del botón de opción. El valor predeterminado es 0x000000 (negro).
symbolDisabledColor	Sample	Color del punto del botón de opción cuando el componente está desactivado. El valor predeterminado es 0x848384 (gris oscuro).

Utilización de aspectos con el componente RadioButton

Se pueden aplicar aspectos al componente RadioButton durante la edición si se modifican los símbolos del componente en la biblioteca. Los aspectos del componente RadioButton se encuentran en la carpeta siguiente de la biblioteca de HaloTheme.fla o SampleTheme.fla: Flash UI Components 2/Themes/MMDefault/RadioButton Assets/States. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

Si hay un botón de opción activado y no seleccionado, al mover el puntero sobre él cambia de aspecto. Cuando se hace clic en un botón de opción no seleccionado, se selecciona y muestra el estado false, correspondiente a no presionado. Si el usuario suelta el ratón, el botón de opción muestra el estado true y el botón de opción seleccionado anteriormente en el grupo recupera su estado false. Si se mueve el puntero fuera del botón de opción al tiempo que se presiona el ratón, el botón de opción recupera el estado false y conserva la selección de entrada.

Si se desactiva un botón de opción o un grupo de botones de opción, muestra su estado desactivado sea cual sea la acción del usuario.

Un componente RadioButton utiliza las propiedades de aspecto siguientes:

Nombre	Descripción
`falseUpIcon`	Estado no seleccionado. El valor predeterminado es `RadioFalseUp`.
`falseDownIcon`	Estado presionado no seleccionado. El valor predeterminado es `RadioFalseDown`.
`falseOverIcon`	Estado de puntero encima no seleccionado. El valor predeterminado es `RadioFalseOver`.
`falseDisabledIcon`	Estado desactivado no seleccionado. El valor predeterminado es `RadioFalseDisabled`.
`trueUpIcon`	Estado seleccionado. El valor predeterminado es `RadioTrueUp`.
`trueDisabledIcon`	Estado desactivado seleccionado. El valor predeterminado es `RadioTrueDisabled`.

Cada uno de estos aspectos se corresponde con el icono que indica el estado de RadioButton. RadioButton no tiene bordes ni fondo.

Para crear símbolos de clip de película para aspectos de RadioButton:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta RadioButton Assets a la biblioteca del documento.
4. Expanda la carpeta RadioButton Assets/States en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo RadioFalseDisabled.
6. Personalice el símbolo como desee.

   Por ejemplo, cambie el color blanco del círculo interior por un gris claro.
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.

   Por ejemplo, repita el cambio de color en el círculo interior del símbolo RadioTrueDisabled.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.

9. Arrastre un componente RadioButton al escenario.

   En este ejemplo, arrastre dos instancias para mostrar los dos nuevos símbolos de aspecto.

10. Defina las propiedades de la instancia de RadioButton como desee.

    En este ejemplo, defina una instancia de RadioButton como seleccionada y utilice ActionScript para definir ambas instancias de RadioButton como desactivadas.

11. Seleccione Control > Probar película.

Clase RadioButton

Herencia MovieClip > Clase UIObject > Clase UIComponent > Clase SimpleButton > Componente Button > RadioButton

Nombre de paquete de ActionScript mx.controls.RadioButton

Las propiedades de la clase RadioButton permiten crear en tiempo de ejecución una etiqueta de texto y definir su posición con respecto al botón de opción. También puede asignar valores de datos a los botones de opción, asignarlos a grupos y seleccionarlos en función de los valores de datos o el nombre de instancia.

Si una propiedad de la clase RadioButton se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente RadioButton utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información sobre la creación de desplazamientos de la selección, consulte "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.RadioButton.version);
```

NOTA El código `trace(myRadioButtonInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase RadioButton

No hay métodos exclusivos de la clase RadioButton.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase RadioButton de la clase UIObject. Al llamar a estos métodos desde el objeto RadioButton, debe utilizarse la forma *RadioButtonInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase RadioButton de la clase UIComponent. Al llamar a estos métodos desde el objeto RadioButton, debe utilizarse la forma *RadioButtonInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase RadioButton

En la tabla siguiente se enumeran las propiedades de la clase RadioButton.

Propiedad	Descripción
`RadioButton.data`	Valor asociado a una instancia de botón de opción.
`RadioButton.groupName`	Nombre de grupo de una instancia de grupo de botones de opción o una instancia de botón de opción.
`RadioButton.label`	Texto que aparece junto al botón de opción.
`RadioButton.labelPlacement`	Orientación del texto de la etiqueta con respecto al botón de opción o al grupo de botones de opción.
`RadioButton.selected`	Selecciona el botón de opción y anula la selección del botón seleccionado anteriormente. Esta propiedad puede utilizarse con una instancia de RadioButton o una instancia de RadioButtonGroup.
`RadioButton.selectedData`	Selecciona el botón de opción con el valor de datos especificado en un grupo de botones de opción.
`RadioButton.selection`	Referencia al botón de opción seleccionado actualmente en un grupo de botones de opción. Esta propiedad puede utilizarse con una instancia de RadioButton o una instancia de RadioButtonGroup.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase RadioButton de la clase UIObject. Al acceder a estas propiedades desde el objeto RadioButton, debe utilizarse la forma *`RadioButtonInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.

Propiedad	Descripción
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase RadioButton de la clase UIComponent. Al acceder a estas propiedades desde el objeto RadioButton, debe utilizarse la forma *RadioButtonInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase SimpleButton

En la tabla siguiente se enumeran las propiedades que hereda la clase RadioButton de la clase SimpleButton. Al acceder a estas propiedades desde el objeto RadioButton, debe utilizarse la forma *RadioButtonInstance.propertyName*.

Propiedad	Descripción
SimpleButton.emphasized	Indica si el botón tiene el aspecto de un botón de comando predeterminado.
SimpleButton.emphasizedStyleDeclaration	Declaración de estilos cuando la propiedad `emphasized` está definida en `true`.

Propiedad	Descripción
SimpleButton.selected	Valor booleano que indica si el botón está seleccionado (true) o no (false). El valor predeterminado es false.
SimpleButton.toggle	Valor booleano que indica si el botón se comporta como un conmutador (true) o no (false). El valor predeterminado es false.

Propiedades heredadas de la clase Button

En la tabla siguiente se enumeran las propiedades que hereda la clase RadioButton de la clase Button. Al acceder a estas propiedades desde el objeto RadioButton, debe utilizarse la forma *RadioButtonInstance.propertyName*.

Propiedad	Descripción
Button.icon	Especifica un icono para una instancia del botón.
Button.label	Especifica el texto que aparece en un botón.
Button.labelPlacement	Especifica la orientación del texto de la etiqueta con respecto a un icono.

Resumen de eventos de la clase RadioButton

En la tabla siguiente se muestra el evento de la clase RadioButton.

Evento	Descripción
RadioButton.click	Se activa cuando se hace clic con el ratón en un botón de opción o un grupo de botones de opción.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase RadioButton de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.

Evento	Descripción
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase RadioButton de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase SimpleButton

En la tabla siguiente se muestra el evento que hereda la clase RadioButton de la clase SimpleButton.

Evento	Descripción
SimpleButton.click	Se difunde cuando se hace clic con el ratón (se suelta el botón del ratón) en un botón o cuando éste está seleccionado y se presiona la barra espaciadora.

RadioButton.click

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object) {
  // ...
};
radioButtonGroup.addEventListener("click", listenerObject);
```

Sintaxis 2:

```
on (click) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se presiona y se suelta) sobre el botón de opción o al seleccionar el botón con las teclas de flecha. El evento también se difunde si se presiona la barra espaciadora o las teclas de flecha cuando la selección está sobre un grupo de botones de opción pero no hay ninguno de ellos seleccionado.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*radioButtonInstance*) distribuye un evento (en este caso, `click`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de RadioButton. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el botón de opción `myRadioButton`, envía "_level0.myRadioButton" al panel Salida:

```
on (click) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se crean tres botones de opción, se colocan en el escenario y se crea un detector para el evento click. Cuando un usuario hace clic en uno de los tres botones de opción, el detector muestra el nombre de instancia del botón de opción seleccionado.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Colocar RadioButtons en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Crear un objeto detector.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The selected radio instance is " + evt_obj.target.selection);
}
// Añadir detector.
radioGroup.addEventListener("click", rbListener);
```

RadioButton.data

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.data

Descripción

Propiedad; especifica los datos que se asociarán con una instancia de RadioButton. La definición de esta propiedad sustituye el valor del parámetro de datos definido durante la edición. La propiedad `data` puede ser cualquier tipo de datos.

Ejemplo

En el siguiente ejemplo se asignan los valores de datos `0xFF00FF` y la etiqueta `#FF00FF` a la instancia de botón de opción `my_rb`. A continuación, se crea un detector para un evento click y se muestra el valor de los datos del botón cuando un usuario hace clic en el botón.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

this.createClassObject(mx.controls.RadioButton, "my_rb", 10,
  {label:"first", groupName:"radioGroup"});

my_rb.data = 0xFF00FF;
my_rb.label = "#FF00FF";

var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The data value for my_rb is " + my_rb.data);
}
// Añadir detector.
my_rb.addEventListener("click", rbListener);
```

RadioButton.groupName

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.groupName

radioButtonGroup.groupName

Descripción

Propiedad; define el nombre de grupo para una instancia o un grupo de botones de opción. Puede utilizar esta propiedad para obtener o definir un nombre de grupo para una instancia de botón de opción o un grupo de botones de opción. Si se llama a este método, se sustituye el valor del parámetro groupName definido durante la edición. El valor predeterminado es `"radioGroup"`.

Ejemplo

En el siguiente ejemplo se establece el nombre de grupo de un grupo de tres botones de opción como `myrbGroup`. Se colocan los botones y, a continuación, se crea un detector para un evento click en el grupo de botones de opción. Cuando el usuario hace clic en un botón de opción, el ejemplo muestra la propiedad `groupName` para el botón en el que se hizo clic.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"myrbGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"myrbGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"myrbGroup"});

// Colocar botones de opción en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Crear un objeto detector.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The selected radio button group name is " +
  evt_obj.target.groupName);
}
// Añadir detector.
myrbGroup.addEventListener("click", rbListener);
```

RadioButton.label

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.label

Descripción

Propiedad; especifica la etiqueta de texto para el botón de opción. De forma predeterminada, la etiqueta aparece a la derecha del botón de opción. Si se llama a este método, se sustituye el parámetro de etiqueta especificado durante la edición. Si el tamaño del texto de la etiqueta supera el del recuadro de delimitación del componente, se recorta el texto. Puede llamar a `RadioButton.setSize()` para aumentar el tamaño del área de etiqueta, pero el texto no se ajusta a la línea siguiente.

Para proporcionar una etiqueta con texto que se ajusta, puede combinar un componente RadioButton sin etiqueta y un componente TextArea para que actúen como un solo RadioButton con ajuste de texto. En el siguiente ejemplo se crea ese botón de opción. Se supone que tiene un componente RadioButton y un componente TextArea en la biblioteca y se desactiva el borde del componente TextArea. La propiedad `label` sería undefined en este caso, si ha accedido a ella.

```
this.createClassObject(mx.controls.RadioButton, "sameas_rb", 1,
  {groupName:"myGroup"});
sameas_rb.move(0,30)
this.createClassObject(mx.controls.TextArea, "message_ta", 2);
message_ta.setSize(200, 60);
// Desactivar el borde del componente TextArea.
message_ta.borderStyle = "none";
message_ta.wordWrap = true;
message_ta.text = "Click here if your shipping information is the same as
  your billing information.";
message_ta.move(20, 30);
```

Ejemplo

En el siguiente ejemplo se crea un botón de opción y se le asigna una etiqueta “Remove from list”.

Primero, añada un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal.

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

this.createClassObject(mx.controls.RadioButton, "my_rb", 10);

// Cambiar el tamaño del componente RadioButton.
my_rb.setSize(200, my_rb.height);
my_rb.label = "Remove from list";
```

RadioButton.labelPlacement

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.labelPlacement

radioButtonGroup.labelPlacement

Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a un botón de opción. Puede definir esta propiedad para una sola instancia o para un grupo de botones de opción. Si la propiedad se define para un grupo, la etiqueta se coloca en la posición apropiada para cada botón del grupo.

A continuación se indican los cuatro valores posibles:

- `"right"` El botón de opción se fija en la esquina superior izquierda del área de delimitación. La etiqueta se coloca a la derecha del botón de opción.
- `"left"` El botón de opción se fija en la esquina superior derecha del área de delimitación. La etiqueta se coloca a la izquierda del botón de opción.
- `"bottom"` La etiqueta se coloca por debajo del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.

- `"top"` La etiqueta se coloca por encima del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.

Ejemplo

El siguiente código crea tres botones de opción y utiliza la propiedad `labelPlacement` para colocar la etiqueta del segundo botón en el lado izquierdo de éste.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal.

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Colocar botones de opción en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

second_rb.labelPlacement = "left";
```

RadioButton.selected

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.selected

radioButtonGroup.selected

Descripción

Propiedad; valor booleano que define el estado de un botón de opción en seleccionado (`true`) y anula la selección de cualquier otro botón seleccionado anteriormente, o bien define el botón de opción en no seleccionado (`false`).

Ejemplo

En el siguiente ejemplo se crean tres botones de opción en un grupo de opciones, se colocan los botones y se establece la propiedad `selected` en `true` para ponerla en el estado seleccionado.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal.

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Colocar botones de opción en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

first_rb.selected = true;
```

RadioButton.selectedData

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonGroup.selectedData

Descripción

Propiedad; selecciona el botón de opción con el valor de datos especificado y anula la selección del botón seleccionado anteriormente. Si no se ha especificado la propiedad `data` de una instancia seleccionada, se selecciona y devuelve el valor de `label` de la instancia seleccionada. La propiedad `selectedData` puede ser cualquier tipo de datos.

Ejemplo

En el siguiente ejemplo se crean tres botones de opción en un grupo de opciones, se colocan los botones y se selecciona el botón cuyo valor de datos es 10, que es el segundo botón.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal.

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first", data:5,
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  data:10, groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  data:15, groupName:"radioGroup"});

// Colocar botones de opción en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

radioGroup.selectedData = 10;
```

RadioButton.selection

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.selection

radioButtonGroup.selection

Descripción

Propiedad; se comporta de forma distinta si la propiedad se obtiene o se define. Si la propiedad se obtiene, devuelve la referencia a un objeto del botón seleccionado en un grupo de botones de opción. Si la propiedad se define, selecciona el botón de opción especificado (que se pasa como una referencia a un objeto) de un grupo de botones de opción y anula la selección del botón seleccionado anteriormente.

Ejemplo

En el siguiente ejemplo se crean tres botones de opción en un grupo de opciones, se colocan los botones y se crea un detector para un evento click del grupo de opciones. Cuando el usuario hace clic en un botón de opción, el detector utiliza la propiedad `selection` para mostrar el nombre de instancia del botón sobre el que se ha hecho clic.

Primero, arrastre un componente RadioButton desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1 de la línea de tiempo principal.

```
/**
 Se requiere:
  - Componente RadioButton en la biblioteca
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Colocar botones de opción en el escenario.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Crear un objeto detector.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
 trace("The selected radio instance is " + radioGroup.selection);
}
// Añadir detector.//
radioGroup.addEventListener("click", rbListener);
```

CAPÍTULO 36

Componente RadioButtonGroup

36

Para más información sobre la clase RadioButtonGroup, consulte Componente RadioButton.

CAPÍTULO 37

Componente RDBMSResolver (sólo en Flash Professional)

37

Los componentes Resolver se utilizan con el componente DataSet (parte de la funcionalidad de administración de datos de la arquitectura de datos de Flash) para guardar los cambios en un origen de datos externo. Son tanto el componente RDBMSResolver como el componente XUpdateResolver. Los componentes Resolver toman un paquete delta (devuelto por `DataSet.deltaPacket`) y lo convierten en un paquete de actualización con un formato apropiado para el tipo de Resolver. Los componentes Connector pueden transmitir a continuación el paquete de actualización al origen de datos externo. Los componentes Resolver no aparecen en pantalla en tiempo de ejecución.

El componente RDBMSResolver crea un paquete de actualización XML que puede analizarse fácilmente en sentencias SQL para actualizar una base de datos relacional. El componente RDBMSResolver se conecta a la propiedad `DeltaPacket` de un componente DataSet, envía su propio paquete de actualización a un conector, recibe los errores de servidor del conector y los vuelve a transferir al componente DataSet, todo ello mediante propiedades que pueden vincularse.

Para más información sobre la arquitectura de datos de Flash, consulte "Resolución de datos (sólo para Flash Professional)" en *Utilización de Flash*. Para más información sobre bases de datos relacionales, consulte "Resolución de datos en una base de datos relacional (sólo para Flash Professional)" en *Utilización de Flash*. Si desea ver un ejemplo completo de una aplicación que actualiza datos mediante el componente RDBMSResolver, consulte www.macromedia.com/devnet/mx/flash/articles/delta_packet.html.

NOTA

Puede utilizar el componente RDBMSResolver para enviar actualizaciones de datos a cualquier objeto que escriba y que pueda analizar XML y generar sentencias SQL para una base de datos (por ejemplo, una página ASP, un servlet Java o un componente de ColdFusion).

Utilización del componente RDBMSResolver (sólo en Flash Professional)

Utilice el componente RDBMSResolver sólo cuando la aplicación Flash contenga un componente Dataset y deba enviar de nuevo una actualización al origen de datos. Este componente resuelve los datos que se desea devolver a una base de datos relacional.

Parámetros de RDBMSResolver

A continuación se indican los parámetros de edición que se pueden definir para cada instancia de RDBMSResolver mediante la ficha Parámetros del Inspector de componentes:

TableName es una cadena que representa **el nombre (en el código XML) de la** tabla de base de datos que se debe actualizar. Esta cadena debería coincidir con el nombre del elemento `RDBMSResolver.fieldInfo` que se debe actualizar. Si no hay actualizaciones en este campo, este parámetro debería estar en blanco, que es el valor predeterminado.

UpdateMode es un enumerador que determina el modo en el que se identifican los campos clave durante la generación del paquete de actualización XML. Los valores posibles son los siguientes:

- `umUsingAll` Utiliza los valores anteriores de todos los campos modificados para identificar el registro que debe actualizarse. Es el valor más seguro que debe utilizarse para realizar actualizaciones, porque garantiza que otro usuario no haya modificado el registro desde que se ha recuperado. Sin embargo, este método es muy lento y genera un paquete de actualización más grande.
- `umUsingModified` Utiliza los valores anteriores de todos los campos modificados para identificar el registro que debe actualizarse. Este valor garantiza que otro usuario no haya modificado los mismos campos del registro desde que se ha recuperado.
- `umUsingKey` Es el valor predeterminado. Esta configuración utiliza el valor anterior de los campos clave. Esto implica un modelo de "simultaneidad optimista" que utiliza la mayor parte de los sistemas de bases de datos actualmente y garantiza que se modifique el mismo registro que se ha recuperado de la base de datos. Los cambios sobrescriben los cambios que los demás usuarios hayan efectuado en los mismos datos.

NullValue es una cadena que representa un valor de campo nulo. Este parámetro se puede personalizar para evitar confundirlo con una cadena vacía (`""`) o con otro valor válido. El valor predeterminado es `{_NULL_}`.

FieldInfo es una colección que representa uno o más campos clave que identifican registros de forma exclusiva. Si el origen de datos es una tabla de base de datos, ésta debe tener uno o más campos que cifren en clave de forma exclusiva los registros que la componen. Además, puede que se hayan calculado o unido campos de otras tablas. Estos campos deben identificarse para que se puedan establecer los campos clave dentro del paquete de actualización XML y para que los campos que no se hayan actualizado se omitan en el paquete de actualización XML.

El parámetro FieldInfo permite utilizar propiedades para designar campos que requieren un manejo especial. Cada elemento de la colección contiene tres propiedades:

- `FieldName` Nombre de un campo. Debe coincidir con un nombre de campo del componente DataSet.
- `OwnerName` Valor opcional que se utiliza para identificar campos que no son "propiedad" de la misma tabla definida en el parámetro TableName del componente RDBMSResolver. Si esta propiedad tiene el mismo valor que el del parámetro TableName o está en blanco, el campo se suele incluir en el paquete de actualización XML. Si tiene otro valor, este campo se excluye del paquete de actualización.
- `IsKey` Propiedad booleana que debe establecerse en `true` para que todos los campos clave de la tabla se actualicen.

En el ejemplo siguiente se muestran los elementos FieldInfo que se crean para actualizar campos en una tabla de clientes. Debe identificar los campos clave en la tabla de clientes. La tabla de clientes tiene un solo campo clave, `id`; por lo tanto, el usuario debe crear un elemento de campo con los valores siguientes:

```
FieldName = "id"
OwnerName = <--! leave this value blank -->
IsKey = "true"
```

Además, el campo `custType` se añade mediante una unión con la consulta. Este campo debe excluirse de la actualización, por lo que se debe crear un elemento de campo con los valores siguientes:

```
FieldName = "custType"
OwnerName = "JoinedField"
IsKey = "false"
```

Cuando se definen elementos de campo, Flash Player puede utilizarlos para generar de forma automática todo el código XML, que se utiliza para actualizar una tabla.

NOTA

El parámetro FieldInfo utiliza una función de Flash llamada editor de colecciones. Cuando se selecciona el parámetro FieldInfo, aparece el cuadro de diálogo del editor de colecciones, donde se pueden añadir nuevos elementos FieldInfo y definir sus propiedades `fieldName`, `ownerName` y `isKey` desde un solo lugar.

Flujo de trabajo normal del componente RDBMSResolver

En los siguientes pasos se describe el flujo de trabajo típico del componente RDBMSResolver.

Para utilizar un componente RDBMSResolver:

1. Añada a la aplicación dos instancias del componente WebServiceConnector y una instancia de cada uno de los componentes DataSet y RDBMSResolver, y asígneles nombres de instancia.
2. Seleccione el primer componente WebServiceConnector. A continuación, utilice la ficha Parámetros del inspector de componentes para introducir la URL del lenguaje de descripción de servicios Web (WSDL, Web Service Definition Language) de un servicio Web que muestre datos desde un origen de datos externo.

NOTA	El servicio Web debe devolver una matriz de registros para vincularse al juego de datos.

3. Utilice la ficha Vinculaciones del Inspector de componentes para vincular la propiedad results del primer componente WebServiceConnector a la propiedad `dataProvider` del componente DataSet.
4. Seleccione el componente DataSet y utilice la ficha Vinculaciones del inspector de componentes para vincular elementos de datos (campos DataSet) a los componentes visuales de su aplicación.
5. Vincule la propiedad `deltaPacket` de DataSet a la propiedad `deltaPacket` de RDBMSResolver.

   Las instrucciones de actualización se envían del componente DataSet al componente RDBMSResolver cuando se llama al método `DataSet.applyUpdates()`.
6. Vincule la propiedad `updatePacket` de RDBMSResolver a la propiedad `params` del segundo componente WebServiceConnector para enviar nuevamente los datos a un método que analizará el paquete de actualización XML. Establezca que la propiedad `params` sea active automáticamente para que el conector envíe el paquete de actualización tan pronto como la vinculación de datos lo copie.
7. Utilice un desencadenante para iniciar la operación de vinculación de datos: utilice el comportamiento Trigger Data Source asociado a un botón o añada ActionScript.

Además de estos pasos, también puede utilizar el componente RDBMSResolver para crear vinculaciones para aplicar el paquete de resultados devuelto desde el servidor al conjunto de datos.

Si desea ver un ejemplo paso a paso de resolución de datos en una base de datos relacional mediante el componente RDBMSResolver, consulte los tutoriales sobre DevNet en http://www.macromedia.com/devnet/mx/flash/data_integration.html.

Clase RDBMSResolver (sólo en Flash Professional)

Herencia MovieClip > RDBMSResolver

Nombre de paquete de ActionScript mx.data.components.RDBMSResolver

Los métodos, propiedades y eventos de la clase RDBMSResolver permiten conectar con un componente DataSet y realizar cambios en orígenes de datos externos.

Resumen de métodos del componente RDBMSResolver

En la tabla siguiente se muestra el método de la clase RDBMSResolver.

Método	Descripción
`RDBMSResolver.addFieldInfo()`	Añade un nuevo elemento a la colección `fieldInfo`, que se utiliza para configurar un componente RDBMSResolver dinámicamente en tiempo de ejecución.

Resumen de propiedades del componente RDBMSResolver

En la tabla siguiente se enumeran las propiedades de la clase RDBMSResolver.

Propiedad	Descripción
`RDBMSResolver.deltaPacket`	La propiedad `deltaPacket` del objeto DataSet debería vincularse a esta propiedad para que, cuando se llame al método `DataSet.applyUpdates()`, se copie a través de la vinculación y el componente Resolver cree el paquete de actualización.
`RDBMSResolver.fieldInfo`	Colección de campos con propiedades que identifican campos DataSet que requieren un manejo especial, ya sea porque son campos clave o porque no pueden actualizarse.
`RDBMSResolver.nullValue`	Cadena que se coloca en el paquete de actualización para indicar que el valor de un campo es `null`.
`RDBMSResolver.tableName`	Identifica la tabla de base de datos que se va a actualizar.

Propiedad	Descripción
`RDBMSResolver.updateMode`	Valores que determinan cómo se identifican los campos clave cuando se genera el paquete de actualización XML.
`RDBMSResolver.updatePacket`	El paquete XML producido por este componente Resolver que contiene los cambios del paquete delta del juego de datos.
`RDBMSResolver.updateResults`	Un paquete delta que contiene los resultados de una actualización devuelta por el servidor a través de un conector.

Resumen de eventos del componente RDBMSResolver

En la tabla siguiente se enumeran los eventos de la clase RDBMSResolver.

Evento	Descripción
`RDBMSResolver.beforeApplyUpdates`	Se define en la aplicación; el componente RDBMSResolver lo llama para realizar modificaciones personalizadas en el código XML de la propiedad `updatePacket` antes de que se vincule al conector.
`RDBMSResolver.reconcileResults`	Se define en la aplicación; el componente RDBMSResolver lo llama para comparar dos paquetes después de recibir los resultados del servidor y de que éstos se hayan aplicado al paquete delta.
`RDBMSResolver.reconcileUpdates`	Se define en la aplicación; el componente RDBMSResolver lo llama cuando se han recibido resultados del servidor después de aplicar las actualizaciones de un paquete delta.

RDBMSResolver.addFieldInfo()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

```
resolveData.addFieldInfo("fieldName", "ownerName", "isKey")
```

Parámetros

fieldName Cadena; proporciona el nombre del campo que este objeto de información describe.

ownerName Cadena; proporciona el nombre de la tabla que es propietaria de este campo. Si este nombre coincide con la propiedad `tableName` de la instancia de RDBMSResolver, puede dejar este parámetro en blanco (`""`).

isKey Valor booleano; indica si este campo es un campo clave.

Valor devuelto

Ninguno.

Descripción

Método; añade un nuevo elemento a la colección `fieldInfo` XML en el paquete de actualización. Utilice este método si debe configurar un componente RDBMSResolver dinámicamente en tiempo de ejecución en lugar de utilizar el Inspector de componentes en el entorno de edición.

Ejemplo

En el ejemplo siguiente se crea un componente RDBMSResolver y se proporciona el nombre de la tabla y el del campo clave, y se impide que el campo `personTypeName` se actualice:

```
var myResolver:RDBMSResolver = new RDBMSResolver();
myResolver.tableName = "Customers";
// Configura el campo id como campo clave
// y el campo personTypeName para que no se actualice.
myResolver.addFieldInfo("id", "", true);
myResolver.addFieldInfo("personTypeName", "JoinedField", false);
// Configura las vinculaciones de datos
//...
```

RDBMSResolver.beforeApplyUpdates

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.beforeApplyUpdates(*eventObject*)

Parámetros

eventObject Objeto de evento Resolver; describe las personalizaciones del paquete XML antes de que se envíe la actualización a través del conector a la base de datos. Este objeto de evento debe contener las propiedades siguientes:

Propiedad	Descripción
target	Objeto; Resolver que activa este evento.
type	Cadena; nombre del evento.
updatePacket	Objeto XML; objeto XML que está a punto de aplicarse.

Valor devuelto

Ninguno.

Descripción

Propiedad; propiedad de tipo `deltaPacket`. Recibe un paquete delta que debe convertirse en un paquete de actualización y emite un paquete delta a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`. Este controlador de eventos proporciona un método para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade nuevamente un delta con mensajes al paquete delta para que se pueda reenviar la próxima vez que el paquete delta se envíe al servidor. Debe escribir código que gestione deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente paquete delta.

Ejemplo

En el ejemplo siguiente se añaden los datos de autenticación de usuario al paquete XML:

```
on (beforeApplyUpdates) {
   // Añadir datos de autenticación de usuarios.
   var userInfo = new XML("" + getUserId() + ""+getPassword() + "");
   updatePacket.firstChild.appendChild(userInfo);
}
```

RDBMSResolver.deltaPacket

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.deltaPacket

Descripción

Propiedad; propiedad de tipo `deltaPacket`. Recibe un paquete delta que debe convertirse en un paquete de actualización y emite un paquete delta a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade nuevamente un delta con mensajes al paquete delta para que se pueda reenviar la próxima vez que el paquete delta se envíe al servidor. Debe escribir código que gestione deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente paquete delta.

RDBMSResolver.fieldInfo

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.fieldInfo

Descripción

Propiedad: especifica una colección de un número ilimitado de campos con propiedades que identifican campos DataSet que requieren un manejo especial, ya sea porque son campos clave o porque no pueden actualizarse (para más información sobre cómo añadir un archivo, consulte `RDBMSResolver.addFieldInfo()`). Cada elemento `fieldInfo` de la colección contiene tres propiedades:

Propiedad	Descripción
`fieldName`	Nombre de campo del caso determinado. Este nombre de campo debe coincidir con un nombre de campo de DataSet.
`ownerName`	Propiedad opcional. Si este campo no es "propiedad" de la tabla definida en la propiedad `RDBMSResolver.tableName`, `OwnerName` es el nombre del propietario de este campo. Si `OwnerName` tiene el mismo valor que `RDBMSResolver.tableName` o está en blanco, el campo se suele incluir en el paquete de actualización XML. Si `OwnerName` no tiene ninguno de estos valores, este campo se excluye del paquete de actualización.
`isKey`	Valor booleano; si su valor es `true`, se actualizan todos los campos clave de la tabla.

RDBMSResolver.nullValue

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`resolveData``.nullValue`

Descripción

Propiedad; cadena que se utiliza para proporcionar un valor null para el valor de un campo. Esta propiedad se puede personalizar para evitar confundirlo con una cadena vacía (`""`) o con otro valor válido. La cadena predeterminada es `{_NULL_}`.

RDBMSResolver.reconcileResults

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.reconcileResults(*eventObject*)

Parámetros

eventObject Objeto de evento Resolver; describe el objeto de evento que se utiliza para comparar dos paquetes de actualización. Este objeto de evento debe contener las propiedades siguientes:

Propiedad	Descripción
target	Objeto; Resolver que difunde este evento.
type	Cadena; nombre del evento.

Valor devuelto

Ninguno.

Descripción

Evento; el componente RDBMSResolver lo difunde para comparar dos paquetes después de recibir los resultados del servidor y de que éstos se hayan aplicado al paquete delta.

Un solo paquete updateResults puede contener los resultados de las operaciones que había en el paquete delta e información sobre actualizaciones realizadas por otros clientes. Cuando se recibe un nuevo paquete de actualización, los resultados de la operación y las actualizaciones de las bases de datos se dividen en dos paquetes de actualización y se colocan por separado en la propiedad deltaPacket. El evento reconcileResults se difunde justo antes de que el paquete delta que contiene los resultados de la operación se envíe mediante la vinculación de datos.

Ejemplo

En el ejemplo siguiente se igualan dos paquetes de actualización y se devuelven y borran las actualizaciones si son correctas:

```
on (reconcileResults) {
  // Examinar resultados.
  if (examine(updateResults)) {
    myDataSet.purgeUpdates();
  } else {
    displayErrors(results);
  }
}
```

RDBMSResolver.reconcileUpdates

Disponibilidad

Flash Player 7.

Edición

Flash Professional 8.

Utilización

resolveData.reconcileUpdates(*eventObject*)

Parámetros

eventObject Objeto de evento Resolver; describe las personalizaciones del paquete XML antes de que la actualización se envíe a través del conector a la base de datos. Este objeto de evento debe contener las propiedades siguientes:

Propiedad	Descripción
target	Objeto; Resolver que difunde este evento.
type	Cadena; nombre del evento.

Valor devuelto

Ninguno.

Descripción

Evento; el componente RDBMSResolver lo difunde cuando se han recibido resultados del servidor después de aplicar las actualizaciones de un paquete delta. Un solo paquete `updateResults` puede contener los resultados de las operaciones que había en el paquete delta e información sobre actualizaciones realizadas por otros clientes. Cuando se recibe un nuevo paquete de actualización, los resultados de la operación y las actualizaciones de las bases de datos se dividen en dos paquetes delta, que se colocan por separado en la propiedad `deltaPacket`. El evento `reconcileUpdates` se difunde justo antes de que el paquete delta que contiene las actualizaciones de base de datos se envíe mediante la vinculación de datos.

Ejemplo

En el ejemplo siguiente se igualan dos resultados y se borran las actualizaciones si han sido correctas:

```
on (reconcileUpdates) {
  // Examinar resultados.
  if (examine(updateResults)) {
    myDataSet.purgeUpdates();
  } else {
    displayErrors(results);
  }
}
```

RDBMSResolver.tableName

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`resolveData``.tableName`

Descripción

Propiedad; cadena que representa en el código XML el nombre de la tabla de base de datos que se debe actualizar. Esta propiedad también determina los campos que se tienen que enviar en el paquete de actualización. Para determinar esto, el componente RDBMSResolver compara el valor de esta propiedad con el valor proporcionado para la propiedad `fieldInfo.ownerName`. Si un campo no tiene ninguna entrada en la propiedad de colección `fieldInfo`, el campo se incluye en el paquete de actualización. Si un campo tiene una entrada en la propiedad de colección `fieldInfo` y el valor de `ownerName` está en blanco o es idéntico al valor de la propiedad `tableName` del componente RDBMSResolver, el campo se incluye en el paquete de actualización. Si un campo tiene una entrada en la propiedad de colección `fieldInfo` y el valor de `ownerName` no está en blanco y es distinto al valor de la propiedad `tableName` del componente RDBMSResolver, el campo no se incluye en el paquete de actualización.

RDBMSResolver.updateMode

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`resolveData``.updateMode`

Descripción

Propiedad; contiene varios valores que determinan cómo se identifican los campos clave cuando se genera el paquete de actualización XML. Esa propiedad puede tener las siguientes cadenas como valores:

Valor	Descripción
`"umUsingAll"`	Utiliza los valores anteriores de todos los campos modificados para identificar el registro que se debe actualizar. Es el valor más seguro que se puede utilizar para realizar actualizaciones, porque garantiza que otro usuario no ha modificado ningún campo del registro desde que lo ha recuperado. Sin embargo, este método es más lento y genera un paquete de actualización de mayor tamaño.

Valor	Descripción
`"umUsingModified"`	Utiliza los valores anteriores de todos los campos modificados para identificar el registro que se debe actualizar. Este valor garantiza que otro usuario no haya modificado los mismos campos del registro desde que se ha recuperado.
`"umUsingKey"`	Valor predeterminado. Esta configuración utiliza el valor anterior de los campos clave. Esto implica un modelo de "simultaneidad optimista" que utiliza la mayor parte de los sistemas de bases de datos actualmente y garantiza que se modifique el mismo registro que se ha recuperado de la base de datos. Sus cambios sobrescriben los cambios realizados por otro usuario sobre los mismos datos.

RDBMSResolver.updatePacket

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.updatePacket

Descripción

Propiedad; propiedad del tipo XML que contiene un paquete XML que se utiliza para vincular una propiedad de conector que transmite de nuevo el paquete de actualización de cambios convertido al servidor para poder actualizar el origen de los datos. Se trata de un documento XML que contiene el paquete de cambios de DataSet.

RDBMSResolver.updateResults

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.updateResults

Descripción

Propiedad; un paquete delta que contiene los resultados de una actualización devuelta por el servidor a través de un conector. Utilice esta propiedad para transmitir errores y datos actualizados del servidor a un juego de datos; por ejemplo, cuando el servidor asigna nuevos ID para un campo asignado automáticamente. Vincule esta propiedad con la propiedad `results` de un conector de manera que pueda recibir los resultados de una actualización y transmitirlos de nuevo al juego de datos.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade nuevamente un delta con mensajes al paquete delta para que se pueda reenviar la próxima vez que el paquete delta se envíe al servidor. Debe escribir código que gestione deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente paquete delta.

CAPÍTULO 38

Clase RectBorder

38

La clase RectBorder se utiliza como borde en la mayoría de los componentes. Cada tema proporciona una implementación independiente de esta clase, pues tiene su propio conjunto de estilos de borde y propiedades compatibles.

La interacción con la clase RectBorder se realiza principalmente a través de la definición de estilos en otros componentes. Por ejemplo, si se incluye un componente List en un documento y se establece la propiedad de estilo `borderStyle`, el componente List crea una clase RectBorder que utiliza el valor de `borderStyle` de la lista. También es posible crear una implementación personalizada de RectBorder para definir el aspecto del borde de todos los componentes que utilizan RectBorder.

La clase RectBorder tiene cuatro estilos de visualización estándar: `none`, `inset`, `outset` y `solid`.

El tema Halo también añade cuatro estilos de visualización especiales que son utilizados por componentes específicos.

Estilo especial	Componente que lo utiliza
`default`	Window
`alert`	Alert
`dropDown`	ComboBox y DateField
`menuBorder`	Menu y MenuBar

El comportamiento y las propiedades de estilo de RectBorder aquí descritos son iguales para todos los componentes que utilizan la clase RectBorder.

Utilización de estilos con la clase RectBorder

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de RectBorder. La instancia de RectBorder utiliza los siguientes estilos:

- `borderCapColor`
- `borderColor`
- `buttonColor`
- `highlightColor`
- `shadowCapColor`
- `shadowColor`
- `themeColor`

Los estilos disponibles en una instancia determinada de RectBorder dependerán del tema que se utilice y del estilo de borde definido en el componente. Si desea ver una demostración interactiva que muestre la relación entre el tema, el estilo de borde y las propiedades de estilo de color disponibles, consulte Utilización de componentes de la Ayuda.

Los cuatro estilos Halo especiales (`default`, `alert`, `dropDown` y `menuBorder`) contienen algunas líneas cuyos colores no pueden definirse a través de estilos. La única forma posible de modificar estos colores es crear un tema personalizado y modificar el código ActionScript correspondiente en la implementación personalizada de RectBorder.

Para establecer un estilo de borde mediante setStyle:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Arrastre un componente TextArea al escenario y asígnele el nombre de instancia `my_ta`.
3. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones:

```
my_ta.setStyle("borderStyle", "alert");
```

> **NOTA** Se puede establecer el borderStyle en "alert" porque se está utilizando el tema predeterminado (Halo). Si está utilizando un tema diferente, entonces, es posible que los cuatro estilos Halo "especiales", incluido "alert", no estén disponibles.

4. Seleccione Control > Probar película para probar el archivo SWF.

Para establecer varios estilos de borde como parámetros del método createClassObject:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.

2. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones:

```
createClassObject(mx.controls.TextArea, "my_ta", 1, {borderStyle:
  "menuBorder", themeColor: "0x990000"});
```

Para más información, consulte `UIObject.createClassObject()`. O, si desea definir varios estilos y aplicarlos a más de una instancia de componente, puede establecer una declaración de estilo nueva que contenga la configuración de estilo y, a continuación, asociar esa declaración de estilo con las instancias de componente (consulte "Definición de estilos personalizados para grupos de componentes" en *Utilización de componentes*).

3. Seleccione Control > Probar película para probar el archivo SWF.

Para establecer un estilo de borde mediante el tema Sample:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Arrastre un componente Button al escenario y asígnele el nombre de instancia **my_btn**.

   También puede crear la instancia con código ActionScript, como se muestra a continuación (asegúrese de que arrastra primero un componente Button a la biblioteca):

```
createClassObject(mx.controls.Button, "my_btn", 1);
```

3. Seleccione Archivo > Importar > Abrir biblioteca externa.
4. Abra el archivo SampleTheme.fla, ubicado en:
   - Windows: \Archivos de programa\Macromedia\Flash 8*idioma*\Configuration\ComponentFLA\
   - Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/
5. En la biblioteca SampleTheme.fla, busque el clip de película de activos de Button en Flash UI Components > Themes > MMDefault > Button Assets > Button Skin y arrástrelo a la biblioteca del documento actual.
6. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript al panel Acciones:

```
my_btn.setStyle("buttonColor", "0xFFFFFF");
my_btn.setStyle("borderStyle", "solid");
my_btn.setStyle("borderColor", "none");
```

> **NOTA** Si tiene previsto establecer varios estilos y necesita mejorar el rendimiento del componente en tiempo de ejecución, puede establecer una declaración de estilo personalizado que contenga esos estilos y, a continuación, asociar la declaración de estilo personalizado con la instancia del componente (consulte "Definición de estilos personalizados para grupos de componentes" en *Utilización de componentes*).

O puede añadir estas configuraciones a createClassObject, de la siguiente manera:

```
createClassObject(mx.controls.Button, "my_btn", 1, {buttonColor:
  "0xFFFFFF", borderStyle: "solid", borderColor: "none"});
```

7. Seleccione Control > Probar película para probar el archivo SWF.

   Observe que incluso con un `"borderColor"` de `"none"`, el botón tiene un borde gris. En este caso, `"none"` no significa transparente, sino un gris neutro.

Creación de una implementación personalizada de RectBorder

La clase RectBorder se utiliza como aspecto de borde en la mayoría de los componentes de ActionScript 2.0. Las implementaciones predeterminadas de los temas Halo y Sample utilizan código ActionScript para dibujar el borde. Una implementación personalizada debe utilizar código ActionScript para registrarse como la implementación de RectBorder y proporcionar funciones de cambio de tamaño, pero puede utilizar tanto código ActionScript como elementos gráficos para representar el aspecto visual.

Cada implementación de RectBorder debe cumplir los siguientes requisitos:

- Debe ampliar mx.skins.RectBorder o una de sus subclases.
- Debe proporcionar un valor de propiedad `offset` o implementar el método `getBorderMetrics` para devolver la información de cambio de tamaño.
- Debe implementar el método `drawBorder()` para dibujar el borde o cambiar su tamaño.
- Debe admitir los cuatro estilos estándar y los cuatro estilos especiales.

  La implementación puede volver a utilizar bordes estándar como bordes especiales, tal y como sucede en el tema Sample.
- Debe registrarse como la implementación de RectBorder.

Registro global de RectBorder

Todos los componentes buscan en una ubicación central una referencia a la clase RectBorder que se utiliza en el documento, `_global.styles.rectBorderClass`. No es posible especificar que un componente individual utilice una implementación de RectBorder distinta. Para personalizar la clase RectBorder en un componente debe utilizarse la propiedad de estilo `borderStyle`.

Ejemplo de implementación personalizada de RectBorder

Las implementaciones de RectBorder proporcionadas tanto por el tema Halo como por el tema Sample utilizan la interfaz API de dibujo de ActionScript para dibujar los bordes de los distintos estilos. En el siguiente ejemplo se muestra cómo crear una implementación personalizada de RectBorder que utilice símbolos gráficos en la visualización.

Para crear una implementación personalizada de RectBorder:

1. Cree una nueva carpeta en la carpeta Classes/mx/skins correspondiente al nombre de paquete personalizado que se va a utilizar para el borde personalizado.

   Para este ejemplo, utilice `myTheme`.

2. Cree un nuevo archivo AS en la nueva carpeta y guárdelo como RectBorder.as.

3. Copie el siguiente código ActionScript en el nuevo archivo AS:

```
import mx.core.ext.UIObjectExtensions;

class mx.skins.myTheme.RectBorder extends mx.skins.RectBorder
{
  static var symbolName:String = "RectBorder";
  static var symbolOwner:Object = RectBorder;
  var className:String = "RectBorder";

#include "../../core/ComponentVersion.as"

   // Todos estos bordes tienen el mismo tamaño: un píxel.
  var offset:Number = 4;

  function init(Void):Void
  {
    super.init();
  }

  function drawBorder(Void):Void
  {
     // Los gráficos se encuentran en la línea de tiempo del símbolo,
     // por lo que sólo es necesario cambiar el tamaño del borde.
     _width = __width;
     _height = __height;
  }

  // Registrar la clase como RectBorder en todos los componentes que se
  utilicen.
  static function classConstruct():Boolean
  {
    UIObjectExtensions.Extensions();
    _global.styles.rectBorderClass = RectBorder;
    _global.skinRegistry["RectBorder"] = true;
    return true;
  }
  static var classConstructed:Boolean = classConstruct();
  static var UIObjectExtensionsDependency = UIObjectExtensions;
}
```

   Si no se utiliza el paquete `myTheme`, deberá modificarse la declaración de clase según convenga.

4. Guarde el archivo AS.
5. Cree un nuevo archivo FLA.
6. Utilice Insertar > Nuevo símbolo para crear un nuevo símbolo de clip de película.
7. Asígnele el nombre `RectBorder`.
8. Si no se muestran los campos avanzados, haga clic en Avanzado.
9. Seleccione Exportar para ActionScript.

   El identificador se rellenará automáticamente como `RectBorder`.
10. En el campo Clase de AS 2.0, establezca el nombre de clase completo de la implementación personalizada de bordes.

    En este ejemplo se utiliza `mx.skins.myTheme.RectBorder`.
11. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
12. Abra el símbolo de RectBorder para editarlo.
13. Dibuje los gráficos del símbolo.

    Por ejemplo, dibuje un cuadrado muy fino sin relleno. Para que el borde personalizado se vea fácilmente, establezca un color de línea rojo brillante.
14. Asegúrese de que los gráficos quedan alineados en la esquina superior izquierda, con las coordenadas *x* e *y* establecidas en (0,0).

    La implementación personalizada de `drawBorder` establecerá la anchura y la altura según los requisitos del componente.
15. Haga clic en Atrás para volver a la línea de tiempo principal.
16. Arrastre al escenario varios componentes que utilicen RectBorder.

    Por ejemplo, arrastre un componente List, TextArea y TextInput al escenario.
17. Seleccione Control > Probar película.

En este ejemplo se crea una implementación de bordes muy simple con colores y gráficos estáticos. No responde a configuraciones diferentes de `borderStyle`; siempre utiliza los mismos gráficos, independientemente del `borderStyle`. Para ejemplos de implementaciones de bordes más completas, revise los ejemplos de los temas Halo y Sample.

CAPÍTULO 39

Clase Screen (sólo en Flash Professional)

39

La clase Screen es la clase base de las pantallas que se crean en el panel Contorno de pantalla de Flash Professional 8. Las pantallas son contenedores de alto nivel para crear aplicaciones y presentaciones. Para ver información general sobre cómo trabajar con pantallas, consulte Capítulo 14, "Trabajo con pantallas (sólo para Flash Professional)" en *Utilización de Flash*.

La clase Screen tiene dos subclases principales: Slide y Form.

La clase Slide proporciona el comportamiento en tiempo de ejecución de las presentaciones de diapositivas. La clase Slide proporciona además capacidad de navegación y secuenciación incorporada, y permite asociar fácilmente transiciones entre diapositivas mediante comportamientos. Los objetos de diapositiva mantienen un "estado" y permiten al usuario avanzar hasta la diapositiva o el estado siguiente o anterior: cuando se muestra la diapositiva siguiente, la anterior se oculta. Para más información sobre la utilización de la clase Slide para controlar las presentaciones de diapositivas, consulte "Clase Slide (sólo en Flash Professional)" en la página 1171.

La clase Form proporciona el entorno en tiempo de ejecución para las aplicaciones de formularios. Los formularios pueden solaparse y ser contenedores de otros componentes o estar contenidos en éstos. A diferencia de las diapositivas, los formularios no proporcionan ninguna capacidad de secuenciación ni de navegación. Para más información, consulte "Clase Form (sólo en Flash Professional)" en la página 761.

La clase Screen proporciona funciones comunes tanto para diapositivas como para formularios.

Las pantallas saben cómo administrar sus elementos secundarios Cada pantalla incorpora una propiedad que contiene una colección formada por una lista de pantallas secundarias de dicha pantalla. Esta colección se determina mediante la jerarquía de pantallas del panel Contorno de pantalla. Las pantallas pueden tener un número cualquiera de elementos secundarios (incluso ninguno) que, a su vez, también pueden tener elementos secundarios.

Las pantallas pueden ocultar y mostrar sus elementos secundarios Puesto que las pantallas son, básicamente, una colección de clips de película anidados, una pantalla puede controlar la visibilidad de sus elementos secundarios. En el caso de las aplicaciones de formularios, todos los elementos secundarios de una pantalla se visualizan a la vez de forma predeterminada; en el caso de las presentaciones de diapositivas, las pantallas individuales se muestran de una en una.

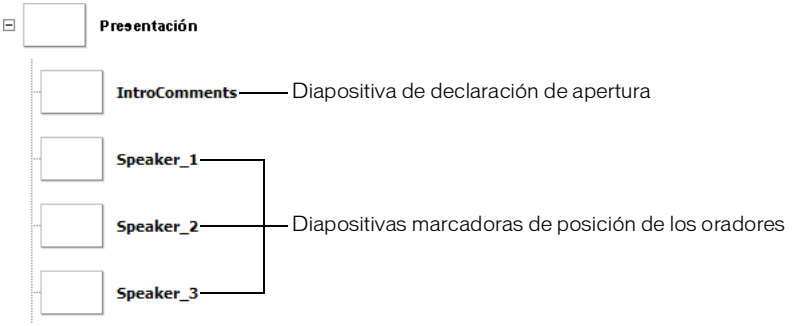
Las pantallas difunden eventos Por ejemplo, puede activar la reproducción de un sonido o empezar a reproducir algún archivo de vídeo cuando se visualice una pantalla concreta.

Carga de contenido externo en pantallas (sólo en Flash Professional)

La clase Screen amplía la clase Loader (véase "Componente Loader" en la página 841), lo cual permite administrar y cargar fácilmente archivos SWF y JPEG externos. La clase Loader contiene una propiedad `contentPath` que especifica la URL de un archivo SWF o JPEG externo o el identificador de vinculación de un clip de película de la biblioteca.

Con esta función, puede cargar un árbol de pantallas externo (o cualquier archivo SWF externo) como elemento secundario de cualquier nodo de pantalla. Esto proporciona una manera útil de separar en módulos los medios basados en pantallas y dividirlos en archivos SWF independientes.

Por ejemplo, suponga que dispone de una presentación de diapositivas en la que tres personas van a aportar, cada una, una sola sección. Puede solicitar a cada orador que cree una presentación de diapositivas independiente (archivo SWF). A continuación puede crear una "presentación de diapositivas maestra" que contenga tres diapositivas marcadoras de posición, una para cada presentación de diapositivas que creen los oradores. Puede hacer que la propiedad `contentPath` de cada diapositiva marcadora de posición señale a uno de los archivos SWF. Por ejemplo, la presentación de diapositivas maestra se puede organizar como se muestra en la ilustración siguiente:



Estructura de la presentación de diapositivas de archivos SWF "maestra"

Suponga que los oradores le proporcionan tres archivos SWF: speaker_1.swf, speaker_2.swf y speaker_3.swf. Puede ensamblar con facilidad la presentación global estableciendo la propiedad `contentPath` de cada diapositiva marcadora de posición, mediante código ActionScript o el inspector de propiedades, tal y como se muestra en el siguiente código:

```
Speaker_1.contentPath = speaker_1.swf;
Speaker_2.contentPath = speaker_2.swf;
Speaker_3.contentPath = speaker_3.swf;
```

De forma predeterminada, cuando se establece la propiedad `contentPath` de una diapositiva durante la edición en el inspector de propiedades, o mediante código ActionScript (como se muestra anteriormente), el archivo SWF especificado se carga inmediatamente después de haberse cargado el archivo SWF de la "presentación maestra". Para reducir el tiempo de carga inicial, considere la posibilidad de establecer la propiedad `contentPath` en un controlador `on(reveal)` asociado con cada diapositiva.

```
// Asociado con la diapositiva Speaker_1
on(reveal) {
  this.contentPath="speaker_1.swf";
}
```

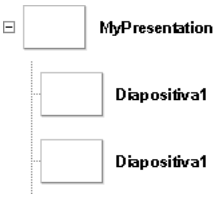
Como alternativa, podría establecer `false` como valor de la propiedad `autoLoad` de la diapositiva. A continuación, podría llamar al método `load()` en la diapositiva cuando se hubiera mostrado la diapositiva. La propiedad `autoLoad` y el método `load()` se heredan de la clase Loader.

```
// Asociado con la diapositiva Speaker_1
on(reveal) {
  this.load();
}
```

Creación de referencias a pantallas cargadas con ActionScript

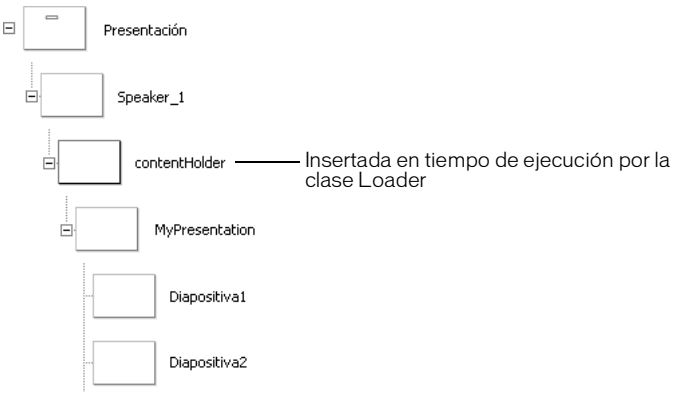
La clase Loader crea un clip de película interno denominado `contentNode` en el que carga el archivo SWF o JPEG especificado por la propiedad `contentPath`. Este clip de película añade, efectivamente, un nodo de pantalla adicional entre la diapositiva "marcadora de posición" (que ha creado en la presentación "maestra" anterior) y la primera diapositiva de la presentación de diapositivas cargada.

Por ejemplo, suponga que el archivo SWF creado para la diapositiva marcadora de posición Speaker_1 (véase la ilustración anterior) tuviera la estructura siguiente, como se muestra en el panel Contorno de pantalla:



Estructura de la presentación de diapositivas de archivos SWF "Speaker 1"

En tiempo de ejecución, cuando el archivo SWF Speaker 1 se carga en la diapositiva marcadora de posición, toda la presentación de diapositivas tiene la estructura siguiente:



Estructura de la presentación "maestra" y "speaker" (ejecución)

Las propiedades y los métodos de las clases Screen, Slide y Form "omiten" este nodo `contentHolder` en la medida de lo posible. Es decir, la diapositiva denominada MyPresentation (y sus subdiapositivas) forma parte del árbol de diapositivas contiguo que nace en la diapositiva Presentation y no se considera un subárbol independiente.

Clase Screen (API) (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > Componente Loader > Screen

Nombre de clase de ActionScript mx.screens.Screen

Los métodos, propiedades y eventos de la clase Screen permiten crear y manipular pantallas en tiempo de ejecución.

Resumen de métodos de la clase Screen

En la tabla siguiente se muestra el método de la clase Screen.

Método	Descripción
`Screen.getChildScreen()`	Devuelve la pantalla secundaria de esta pantalla en un índice concreto.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Screen de la clase UIObject. Al llamar a estos métodos desde el objeto Screen, debe utilizarse la forma *`ScreenInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.

Método	Descripción
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Screen de la clase UIComponent. Al llamar a estos métodos desde el objeto Screen, debe utilizarse la forma *`ScreenInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Métodos heredados de la clase Loader

En la tabla siguiente se enumeran los métodos que hereda la clase Screen de la clase Loader. Al llamar a este método desde el objeto Screen, debe utilizarse la forma *`ScreenInstance.methodName`*.

Método	Descripción
`Loader.load()`	Carga el contenido especificado en la propiedad `contentPath`.

Resumen de propiedades de la clase Screen

En la tabla siguiente se enumeran las propiedades de la clase Screen.

Propiedad	Descripción
`Screen.currentFocusedScreen`	Sólo lectura; devuelve la pantalla que contiene la selección actual global.
`Screen.indexInParent`	Sólo lectura; devuelve el índice de la pantalla (basado en cero) en la lista de pantallas secundarias de su pantalla principal.
`Screen.numChildScreens`	Sólo lectura; devuelve el número de pantallas secundarias que contiene la pantalla.

Propiedad	Descripción
`Screen.parentIsScreen`	Sólo lectura; devuelve un valor booleano (`true` o `false`) que indica si el objeto principal de la pantalla también es una pantalla.
`Screen.parentScreen`	Sólo lectura; devuelve la pantalla que contiene la pantalla especificada.
`Screen.rootScreen`	Sólo lectura; devuelve la pantalla raíz del árbol o subárbol que contiene la pantalla.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Screen de la clase UIObject. Al acceder a estas propiedades desde el objeto Screen, debe utilizarse la forma *`ScreenInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.

Propiedad	Descripción
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Screen de la clase UIComponent. Al acceder a estas propiedades desde el objeto Screen, debe utilizarse la forma *ScreenInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase Loader

En la tabla siguiente se enumeran las propiedades que hereda la clase Screen de la clase Loader. Al acceder a estas propiedades desde el objeto Screen, debe utilizarse la forma *ScreenInstance.propertyName*.

Propiedad	Descripción
Loader.autoLoad	Valor booleano que indica si el contenido se carga automáticamente (true) o si es preciso llamar a Loader.load() (false).
Loader.bytesLoaded	Propiedad de sólo lectura que indica el número de bytes que se han cargado.
Loader.bytesTotal	Propiedad de sólo lectura que indica el número total de bytes del contenido.
Loader.content	Referencia al contenido del componente Loader. Es una propiedad de sólo lectura.
Loader.contentPath	Cadena que indica la URL del contenido que va a cargarse.

Propiedad	Descripción
Loader.percentLoaded	Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.
Loader.scaleContent	Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader (true) o si el componente Loader se redimensiona para adaptarse al contenido (false).

Resumen de eventos de la clase Screen

En la tabla siguiente se enumeran los eventos de la clase Screen.

Evento	Descripción
Screen.allTransitionsInDone	Se difunde cuando todas las transiciones de entrada aplicadas a una pantalla han finalizado.
Screen.allTransitionsOutDone	Se difunde cuando todas las transiciones de salida aplicadas a una pantalla han finalizado.
Screen.mouseDown	Se difunde cuando se ha presionado el botón del ratón sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
Screen.mouseDownSomewhere	Se difunde cuando se ha presionado el botón del ratón en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.
Screen.mouseMove	Se difunde cuando el ratón se mueve mientras se encuentra sobre una pantalla.
Screen.mouseOut	Se difunde cuando el ratón se mueve del interior al exterior de la pantalla.
Screen.mouseOver	Se difunde cuando el ratón se mueve del exterior al interior de la pantalla.
Screen.mouseUp	Se difunde cuando el botón del ratón se ha soltado sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
Screen.mouseUpSomewhere	Se difunde cuando el botón del ratón se ha soltado en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Screen de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Screen de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase Loader

En la tabla siguiente se enumeran los eventos que hereda la clase Screen de la clase Loader.

Evento	Descripción
`Loader.complete`	Se activa cuando el contenido termina de cargarse.
`Loader.progress`	Se activa mientras se carga el contenido.

Screen.allTransitionsInDone

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(allTransitionsInDone) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.allTransitionsInDone = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("allTransitionsInDone", listenerObject)
```

Descripción

Evento; se difunde cuando todas las transiciones de entrada aplicadas a esta pantalla han finalizado. El administrador de transiciones asociado con *screenObj* difunde el evento `allTransitionsInDone`.

Ejemplo

En el ejemplo siguiente, un botón (`nextSlide_btn`) contenido en la diapositiva `mySlide` se visualiza después de que todas las transiciones de entrada aplicadas a `mySlide` hayan finalizado.

```
// Asociado con mySlide:
on(allTransitionsInDone) {
  this.nextSlide_btn._visible = true;
}
```

Véase también

`Screen.allTransitionsOutDone`

Screen.allTransitionsOutDone

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(allTransitionsOutDone) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.allTransitionsOutDone = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("allTransitionsOutDone", listenerObject)
```

Descripción

Evento; se difunde cuando todas las transiciones de salida aplicadas a la pantalla han finalizado. El administrador de transiciones asociado con *screenObj* difunde el evento `allTransitionsOutDone`.

Véase también

`Screen.currentFocusedScreen`

Screen.currentFocusedScreen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myScreen`.currentFocusedScreen`

Descripción

Propiedad estática (sólo lectura); devuelve una referencia al objeto Screen "última diapositiva hoja" que contiene la selección actual global. *Última diapositiva hoja* se refiere a la pantalla más alejada de la pantalla raíz en la jerarquía de pantallas. La selección puede estar en la propia pantalla o en un clip de película, objeto de texto o componente dentro de esa pantalla. Esta propiedad se establece de forma predeterminada en `null` si no hay una selección actual.

Por ejemplo, suponga que tiene una jerarquía de pantallas en tiempo de ejecución similar a la siguiente:

```
presentation
  screen1
    subscreen1_1
      mymovieclip
        myUIButton

  screen2
    subscreen1_2
```

Si `myUIButton` está seleccionado, la pantalla correspondiente a la última diapositiva hoja que contiene la selección es `subscreen1_1`, que es lo que devolvería `currentFocusedScreen`. En este caso, `presentation`, `screen1` y `subscreen1_1` contienen la selección pero el "más cercano" a las hojas del árbol en la jerarquía de pantallas y, por lo tanto, más alejado de la raíz es `subscreen1_1`.

Ejemplo

En el ejemplo siguiente se muestra el nombre de la pantalla seleccionada actualmente en el panel Salida.

```
var currentFocus:mx.screens.Screen =
  mx.screens.Screen.currentFocusedScreen;
trace("Current screen is: " + currentFocus._name);
```

Screen.getChildScreen()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`myScreen``.getChildScreen(`*`childIndex`*`)`

Parámetros

childIndex Número que indica el índice basado en cero de la pantalla secundaria que debe devolverse.

Valor devuelto

Un objeto Screen.

Descripción

Método; devuelve la pantalla secundaria de *myScreen* cuyo índice es *childIndex*.

Ejemplo

El ejemplo siguiente envía al panel Salida los nombres de todas las pantallas secundarias que pertenecen a la pantalla raíz denominada `Presentation`.

```
for (var i:Number = 0; i < _root.Presentation.numChildScreens; i++) {
    var childScreen:mx.screens.Screen =
  _root.Presentation.getChildScreen(i);
    trace(childScreen._name);
}
```

Screen.indexInParent

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myScreen`.indexInParent`

Descripción

Propiedad (sólo lectura); contiene el índice basado en cero de *myScreen* en la lista de pantallas secundarias de su pantalla principal.

Ejemplo

En el ejemplo siguiente se muestra la posición relativa de la pantalla `myScreen` en la lista de pantallas secundarias de su pantalla principal.

```
var numChildren:Number = myScreen._parent.numChildScreens;
var myIndex:Number = myScreen.indexInParent;
trace("I'm child slide # " + myIndex + " out of " + numChildren + "
  screens.");
```

Screen.mouseDown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseDown) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseDown = function(eventObj){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseDown", listenerObject)
```

Descripción

Evento; se difunde cuando se ha presionado el botón del ratón sobre un objeto (por ejemplo, una forma o un clip de película) que es propiedad directa de la pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

El código siguiente muestra el nombre de la pantalla que ha capturado el evento de ratón en el panel Salida.

```
on(mouseDown) {
  trace("Mouse down event on: " + eventObj.target._name);
}
```

Screen.mouseDownSomewhere

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseDownSomewhere) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseDownSomewhere = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseDownSomewhere", listenerObject)
```

Descripción

Evento; se difunde cuando se presiona el botón del ratón, pero no necesariamente en la pantalla especificada.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Screen.mouseMove

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseMove) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseMove = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseMove", listenerObject)
```

Descripción

Evento; se difunde cuando el ratón se mueve mientras se encuentra sobre la pantalla. Este evento sólo se envía cuando el ratón se encuentra sobre el recuadro de delimitación de esta pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

NOTA

Este evento puede afectar el rendimiento del sistema y debería utilizarse con prudencia.

Screen.mouseOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseOut) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseOut = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseOut", listenerObject)
```

Descripción

Evento; se difunde cuando el ratón se mueve del interior del recuadro de límite de la pantalla al exterior de éste.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

NOTA

Este evento puede afectar el rendimiento del sistema y debería utilizarse con prudencia.

Screen.mouseOver

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseOver) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseOver = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseOver", listenerObject)
```

Descripción

Evento; se difunde cuando el ratón se mueve del exterior del recuadro de delimitación de la pantalla al interior de éste.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

NOTA	Este evento puede afectar el rendimiento del sistema y debería utilizarse con prudencia.

Screen.mouseUp

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseUp) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseUp = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseUp", listenerObject)
```

Descripción

Evento; se difunde cuando se suelta el ratón sobre la pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Screen.mouseUpSomewhere

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(mouseUpSomewhere) {
  // El código se escribe aquí.
}
listenerObject = new Object();
listenerObject.mouseUpSomewhere = function(eventObject){
  // Introducir aquí el código propio.
}
screenObj.addEventListener("mouseUpSomewhere", listenerObject)
```

Descripción

Evento; se difunde cuando se suelta el botón del ratón, pero no necesariamente en la pantalla especificada.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObj*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

Screen.numChildScreens

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myScreen.numChildScreens

Descripción

Propiedad (sólo lectura); devuelve el número de pantallas secundarias que contiene *myScreen*.

Ejemplo

En el ejemplo siguiente se muestran los nombres de todas las pantallas secundarias que pertenecen a myScreen.

```
var howManyKids:Number = myScreen.numChildScreens;
for(i=0; i<howManyKids; i++) {
  var childScreen = myScreen.getChildScreen(i);
  trace(childScreen._name);
}
```

Véase también

Screen.getChildScreen()

Screen.parentIsScreen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myScreen.parentIsScreen

Descripción

Propiedad (sólo lectura): devuelve un valor booleano que indica si el objeto principal de la pantalla especificada también es una pantalla (true) o no (false). Si el valor de esta propiedad es false, *myScreen* está en la raíz de su jerarquía de pantallas.

Ejemplo

El código siguiente determina si el objeto principal de la pantalla `myScreen` también es una pantalla. Si el valor de `myScreen.parentIsScreen` es `true`, una sentencia `trace()` muestra el número de diapositivas del mismo nivel de `myScreen` en el panel Salida. Si la pantalla principal de `myScreen` no es una pantalla, Flash supone que `myScreen` es la diapositiva raíz (maestra) de la presentación y que, por lo tanto, no tiene dispositivas del mismo nivel.

```
if (myScreen.parentIsScreen) {
  trace("I have "+myScreen._parent.numChildScreens+" sibling screens");
} else {
  trace("I am the root screen and have no siblings");
}
```

Screen.parentScreen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myScreen`.parentScreen`

Descripción

Propiedad (sólo lectura); devuelve la pantalla que contiene *myScreen*. Devuelve `null` si *myScreen* es la pantalla raíz.

Ejemplo

En el ejemplo siguiente se muestra el nombre de la pantalla que contiene la pantalla `myScreen`.

```
var myParent:mx.screens.Screen = myScreen.rootScreen;
```

Screen.rootScreen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

myScreen.rootScreen

Descripción

Propiedad (sólo lectura); devuelve la pantalla situada en la parte superior de la jerarquía de pantallas que contiene *myScreen*.

Ejemplo

En el ejemplo siguiente se muestra el nombre de la pantalla raíz que contiene la pantalla myScreen.

```
var myRoot:mx.screens.Screen = myScreen.rootScreen;
```

CAPÍTULO 40

Componente ScrollPane

40

El componente ScrollPane muestra clips de película, archivos JPEG y archivos SWF en un área desplazable. El panel de desplazamiento permite limitar la cantidad de área de pantalla que ocupan estos tipos de medios. El panel de desplazamiento puede desplazar contenido cargado desde un disco local o desde Internet. Dicho contenido puede establecerse mediante código ActionScript durante la edición y en tiempo de ejecución.

Cuando el panel de desplazamiento está seleccionado, se seleccionarán los marcadores si el contenido tiene tabulaciones válidas. Después de la última tabulación del contenido, la selección se desplazará al componente siguiente. La selección no se dirige nunca a las barras de desplazamiento vertical y horizontal del panel de desplazamiento.

La instancia de ScrollPane se selecciona cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando la selección esté sobre la instancia de ScrollPane, podrá controlarla con las teclas siguientes:

Tecla	Descripción
Flecha abajo	Mueve el contenido una línea de desplazamiento vertical hacia arriba.
Fin	Mueve el contenido hasta la parte inferior del panel de desplazamiento.
Flecha izquierda	Mueve el contenido una línea de desplazamiento horizontal a la derecha.
Inicio	Mueve el contenido hasta la parte superior del panel de desplazamiento.
Av Pág	Mueve el contenido una página de desplazamiento vertical hacia arriba.
Re Pág	Mueve el contenido una página de desplazamiento vertical hacia abajo.
Flecha derecha	Mueve el contenido una línea de desplazamiento horizontal a la izquierda.
Flecha arriba	Mueve el contenido una línea de desplazamiento vertical hacia abajo.

Para más información sobre el control de la selección, consulte “Clase FocusManager” en la página 745 o “Creación de un desplazamiento personalizado de la selección” en *Utilización de componentes*.

La previsualización dinámica de cada instancia de ScrollPane refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes.

Utilización del componente ScrollPane

Puede utilizar un panel de desplazamiento para mostrar cualquier contenido que sea más grande que el área en la que se ha cargado. Por ejemplo, si tiene una imagen grande que debe mostrarse en un espacio reducido de una aplicación, puede cargarla en un panel de desplazamiento.

Para definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido en el panel, defina el parámetro `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

Los componentes como Loader, ScrollPane y Window tienen eventos para determinar que ha finalizado la carga del contenido. De este modo, si desea establecer propiedades en el contenido de un componente Loader, ScrollPane o Window, añada la sentencia de propiedad en un controlador de eventos “complete”. Observe el siguiente ejemplo:

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._width= 100;
}
my_scrollpane.addEventListener("complete", loadtest)
```

Para más información, consulte “ScrollPane.content” en la página 1143.

Parámetros de ScrollPane

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ScrollPane en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

contentPath indica el contenido que se va a cargar en el panel de desplazamiento. Este valor puede ser una ruta relativa a un archivo SWF o JPEG local, o bien una ruta relativa o absoluta a un archivo en Internet. También puede ser el identificador de vinculación de un símbolo de clip de película de la biblioteca definido en Exportar para ActionScript.

hLineScrollSize indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se hace clic en un botón de flecha. El valor predeterminado es 5.

hPageScrollSize indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se hace clic en la guía de deslizamiento. El valor predeterminado es 20.

hScrollPolicy muestra las barras de desplazamiento horizontal. El valor puede ser `on`, `off` o `auto`. El valor predeterminado es `auto`.

scrollDrag es un valor booleano que determina si se produce el desplazamiento (`true`) o no (`false`) cuando un usuario arrastra el contenido en el panel de desplazamiento. El valor predeterminado es `false`.

vLineScrollSize indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se hace clic en una flecha de desplazamiento. El valor predeterminado es 5.

hPageScrollSize indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se hace clic en la guía de la barra de desplazamiento. El valor predeterminado es 20.

vScrollPolicy muestra las barras de desplazamiento vertical. El valor puede ser `on`, `off` o `auto`. El valor predeterminado es `auto`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente ScrollPane en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes ScrollPane utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase ScrollPane" en la página 1136.

Creación de aplicaciones con el componente ScrollPane

En el procedimiento siguiente se explica cómo añadir un componente ScrollPane a una aplicación durante la edición. En este ejemplo, el panel de desplazamiento carga una foto desde una ruta especificada mediante la propiedad `contentPath`.

Para crear una aplicación con el componente ScrollPane:

1. Arrastre el componente ScrollPane desde el panel Componentes al escenario.
2. En el inspector de propiedades, introduzca el nombre de instancia **my_sp**.
3. Seleccione el fotograma 1 de la línea de tiempo principal, abra el panel Acciones e introduzca el código siguiente:

```
/**
 Se requiere:
  - ScrollPane en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(320, 240);

// Crear objeto detector para posición vertical de desplazamiento.
var scrollListener:Object = new Object();
scrollListener.scroll = function(evt_obj:Object) {
 trace("hPosition: " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Añadir detector.
my_sp.addEventListener("scroll", scrollListener);

// Crear un objeto detector para carga completada.
var completeListener:Object = new Object();
completeListener.complete = function(evt_obj:Object) {
 trace(evt_obj.target.contentPath + " has completed loading.");
};
// Añadir detector.
my_sp.addEventListener("complete", completeListener);

my_sp.contentPath = "http://www.helpexamples.com/flash/images/
  image1.jpg";
```

Los ejemplos crean un panel de desplazamiento, establecen su tamaño y cargan una imagen en él mediante la propiedad `contentPath`. También se crean dos detectores. El primero detecta un evento scroll y muestra la posición de la imagen mientras el usuario se desplaza vertical u horizontalmente. El segundo detecta un evento complete y muestra un mensaje en el panel Salida que indica que la imagen ha terminado de cargarse.

Personalización del componente ScrollPane

El componente ScrollPane puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o cualquier método o propiedad aplicable de la clase ScrollPane.

Debe considerar los puntos siguientes, relacionados con los componentes ScrollPane:

- El componente ScrollPane coloca el punto de registro de su contenido en la esquina superior izquierda del panel.
- Cuando se desactiva la barra de desplazamiento horizontal, la barra vertical se muestra de arriba abajo sobre el lado derecho del panel de desplazamiento. Si se desactiva la barra de desplazamiento vertical, la barra horizontal se muestra de izquierda a derecha sobre el borde inferior del panel de desplazamiento. También se pueden desactivar las dos barras de desplazamiento.
- Si el panel de desplazamiento es demasiado pequeño, es posible que el contenido no se muestre correctamente.
- Cuando se cambia el tamaño del panel de desplazamiento, el tamaño de los botones no se modifica. La guía y el cuadro de desplazamiento (deslizador) se expanden o se contraen, y cambia el tamaño de sus áreas activas.

Utilización de estilos con el componente ScrollPane

ScrollPane admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`borderStyle`	Ambos	El componente ScrollPane utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase “Clase RectBorder” en la página 1103. El estilo de borde predeterminado es `"inset"`.
`scrollTrackColor`	Sample	Color de fondo de la guía de desplazamiento. El valor predeterminado es 0xCCCCCC (gris claro).

Estilo	Tema	Descripción
`symbolColor`	Sample	Color de las flechas de los botones de la barra de desplazamiento. El valor predeterminado es 0x000000 (negro).
`symbolDisabledColor`	Sample	Color de las flechas desactivadas de los botones de la barra de desplazamiento. El valor predeterminado es 0x848384 (gris oscuro).

Utilización de aspectos con el componente ScrollPane

El componente ScrollPane utiliza una instancia de RectBorder para su borde y barras de desplazamiento para los elementos de desplazamiento. Para más información sobre la aplicación de aspectos en estos elementos visuales, consulte "Clase RectBorder" en la página 1103 y "Utilización de aspectos con el componente UIScrollBar" en la página 1435.

Clase ScrollPane

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > ScrollPane

Nombre de clase de ActionScript mx.containers.ScrollPane

Las propiedades de la clase ScrollPane permiten realizar lo siguiente en tiempo de ejecución: definir el contenido, controlar el progreso de carga y ajustar la cantidad de desplazamiento.

Si una propiedad de la clase ScrollPane se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

Puede definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido del panel. Para ello, establezca la propiedad `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.containers.ScrollPane.version);
```

NOTA	El código `trace(myScrollPaneInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase ScrollPane

En la tabla siguiente se enumeran los métodos de la clase ScrollPane.

Método	Descripción
`ScrollPane.getBytesLoaded()`	Devuelve el número de bytes del contenido cargado.
`ScrollPane.getBytesTotal()`	Devuelve el número total de bytes del contenido que se va a cargar.
`ScrollPane.refreshPane()`	Vuelve a cargar el contenido del panel de desplazamiento (pero no redibuja la barra de desplazamiento).

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase ScrollPane de la clase UIObject. Al llamar a estos métodos desde el objeto ScrollPane, debe utilizarse la forma *`ScrollPaneInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.

Método	Descripción
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase ScrollPane de la clase UIComponent. Al llamar a estos métodos desde el objeto ScrollPane, debe utilizarse la forma *`ScrollPaneInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase ScrollPane

En la tabla siguiente se enumeran las propiedades de la clase ScrollPane.

Método	Descripción
`ScrollPane.content`	Referencia al contenido cargado en el panel de desplazamiento (sólo lectura).
`ScrollPane.contentPath`	Cadena que indica una URL absoluta o relativa del archivo SWF o JPEG que se va a cargar en el panel de desplazamiento, o identificador de vinculación de un clip de película en el panel Biblioteca del documento actual.
`ScrollPane.hLineScrollSize`	Cantidad de contenido que se va a desplazar horizontalmente cuando se hace clic en una flecha de desplazamiento.
`ScrollPane.hPageScrollSize`	Cantidad de contenido que se va a desplazar horizontalmente cuando se hace clic en la guía de desplazamiento.
`ScrollPane.hPosition`	Posición horizontal en píxeles de la barra de desplazamiento horizontal del panel de desplazamiento.

Método	Descripción
`ScrollPane.hScrollPolicy`	Estado de la barra de desplazamiento horizontal. Puede estar siempre activada (`"on"`), siempre desactivada (`"off"`), o bien activarse cuando sea necesario (`"auto"`). El valor predeterminado es `"auto"`.
`ScrollPane.scrollDrag`	Indica si se produce el desplazamiento (`true`) o no (`false`) cuando un usuario arrastra contenido en el panel de desplazamiento. El valor predeterminado es `false`.
`ScrollPane.vLineScrollSize`	Cantidad de contenido que se va a desplazar verticalmente cuando se hace clic en una flecha de desplazamiento.
`ScrollPane.vPageScrollSize`	Cantidad de contenido que se va a desplazar verticalmente cuando se hace clic en la guía de desplazamiento.
`ScrollPane.vPosition`	Posición en píxeles de la barra de desplazamiento vertical del panel de desplazamiento.
`ScrollPane.vScrollPolicy`	Estado de la barra de desplazamiento vertical. Puede estar siempre activada (`"on"`), siempre desactivada (`"off"`), o bien activarse cuando sea necesario (`"auto"`). El valor predeterminado es `"auto"`.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase ScrollPane de la clase UIObject. Al acceder a estas propiedades desde el objeto ScrollPane, debe utilizarse la forma *`ScrollPaneInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase ScrollPane de la clase UIComponent. Al acceder a estas propiedades desde el objeto ScrollPane, debe utilizarse la forma *`ScrollPaneInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase ScrollPane

En la tabla siguiente se enumeran los eventos de la clase ScrollPane.

Evento	Descripción
`ScrollPane.complete`	Se difunde cuando se carga el contenido del panel de desplazamiento.
`ScrollPane.progress`	Se difunde mientras se carga el contenido del panel de desplazamiento.
`ScrollPane.scroll`	Se difunde al hacer clic en la barra de desplazamiento.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase ScrollPane de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase ScrollPane de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

ScrollPane.complete

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object) {
  // ...
};
scrollPaneInstance.addEventListener("complete", listenerObject);
```

Sintaxis 2:

```
on (complete) {
  //...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando finaliza la carga del contenido.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia del componente (*scrollPaneInstance*) distribuye un evento (en este caso, `complete`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de ScrollPane. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myScrollPaneComponent` de ScrollPane, envía "_level0.myScrollPaneComponent" al panel Salida:

```
on (complete) {
  trace(this);
}
```

Ejemplo

En el ejemplo siguiente se crea un objeto detector con un controlador de eventos `complete` para la instancia `scrollPane`. Cuando se carga el contenido del panel de desplazamiento, el detector muestra un mensaje en el panel Salida.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - ScrollPane en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(320, 240);

// Crear un objeto detector para carga completada.
var completeListener:Object = new Object();
completeListener.complete = function(evt_obj:Object) {
 trace(evt_obj.target.contentPath + " has completed loading.");
};
// Añadir detector.
my_sp.addEventListener("complete", completeListener);

my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.content

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.content

Descripción

Propiedad de sólo lectura; referencia al contenido del panel de desplazamiento. El valor es `undefined` hasta que comienza la carga.

Ejemplo

En este ejemplo se define la propiedad `contentPath` para cargar un panel de desplazamiento con una foto (o, técnicamente, un clip de película que contiene una imagen JPEG). También se crea un control numérico en el que el usuario puede producir un incremento o decremento de 10, hasta un valor de 100. Cuando el usuario cambia el valor del NumericStepper, un detector establece la transparencia (`content._alpha`) de la imagen en el porcentaje especificado. Observe que `_alpha` es una propiedad de MovieClip.

Primero, arrastre el componente ScrollPane y el componente NumericStepper desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - ScrollPane en la biblioteca
  - NumericStepper en la biblioteca
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.NumericStepper, "my_nstep", 10,
  {minimum:10, maximum:100, stepSize:10});
my_nstep.value = my_nstep.maximum;

this.createClassObject(mx.containers.ScrollPane, "my_sp", 20);
my_sp.move(0, 30);
my_sp.setSize(180, 160);
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image2.jpg";

var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 my_sp.content._alpha = my_nstep.value;
}
my_nstep.addEventListener("change", nstepListener);
```

Véase también

`ScrollPane.contentPath`

ScrollPane.contentPath

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.contentPath

Descripción

Propiedad; cadena que indica una URL absoluta o relativa del archivo SWF o JPEG que se carga en el panel de desplazamiento. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido.

Si se carga el contenido utilizando una URL relativa, el contenido cargado deberá ser relativo a la ubicación del archivo SWF que contiene el panel de desplazamiento. Por ejemplo, una aplicación que utilice un componente ScrollPane situado en el directorio /scrollpane/nav/example.swf podría cargar el contenido del directorio /scrollpane/content/flash/logo.swf con la siguiente propiedad `contentPath`: `"../content/flash/logo.swf"`

Ejemplo

En el siguiente ejemplo se muestra cómo definir la propiedad `contentPath` para cargar un ScrollPane desde tres orígenes distintos: 1) una imagen de Internet; 2) un clip de película de la biblioteca; 3) un archivo SWF del directorio de trabajo actual. Utilice sólo un origen cada vez.

Arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual. Para intentar la opción 2, debe crear un clip de película en la biblioteca y hacer referencia a su nombre. Para intentar la opción 3, cree un archivo SWF en el directorio de trabajo actual y especifique su nombre. A continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - ScrollPane en el escenario (nombre de instancia: my_sp)
  - Símbolo con identificación de vinculación de "movieClip_Name" en la
  biblioteca ** opcional
  - Archivo logo.swf en el directorio de trabajo ** opcional
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_sp:mx.containers.ScrollPane;

// método 1: Imagen JPEG
my_sp.contentPath ="http://www.helpexamples.com/flash/images/image1.jpg";

// método 2: Símbolo en la biblioteca
my_sp.contentPath ="movieClip_Name";

// método 3: Archivo SWF
my_sp.contentPath ="logo.swf";
```

Véase también

ScrollPane.content

ScrollPane.getBytesLoaded()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.getBytesLoaded()

Parámetros

Ninguno.

Valor devuelto

Número de bytes cargados en el panel de desplazamiento.

Descripción

Método; devuelve el número de bytes cargados en la instancia de ScrollPane. Mientras se carga el contenido, invoque este método en intervalos regulares para comprobar el progreso de la operación.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp` y se define un objeto detector denominado `loadListener` con un controlador de eventos `progress`. El controlador de eventos llama a las funciones `getBytesLoaded()` y `getBytesTotal()` para mostrar el progreso de la carga en el panel Salida.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var loadListener:Object = new Object();
loadListener.progress = function(evt_obj:Object) {
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded.");
};
my_sp.addEventListener("progress", loadListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.getBytesTotal()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.getBytesTotal()

Parámetros

Ninguno.

Valor devuelto

Un número.

Descripción

Método; devuelve el número total de bytes que se va a cargar en la instancia de ScrollPane.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp` y se define un objeto detector denominado `loadListener` con un controlador de eventos `progress`. El controlador de eventos llama a las funciones `getBytesLoaded()` y `getBytesTotal()` para mostrar el progreso de la carga en el panel Salida.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var loadListener:Object = new Object();
loadListener.progress = function(evt_obj:Object) {
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded.");
};
my_sp.addEventListener("progress", loadListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

Véase también

`ScrollPane.getBytesLoaded()`

ScrollPane.hLineScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollPaneInstance``.hLineScrollSize`

Descripción

Propiedad; número de píxeles que se va a mover el contenido al hacer clic en una flecha de la barra de desplazamiento horizontal. El valor predeterminado es 5.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se establece la propiedad `hLineScrollSize` para desplazarse 100 píxeles cuando el usuario haga clic en una flecha de la barra de desplazamiento horizontal.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Desplazar 100 píxeles cuando se haga clic en las flechas de la barra
  horizontal.
my_sp.hLineScrollSize = 100;
```

ScrollPane.hPageScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hPageScrollSize

Descripción

Propiedad; número de píxeles que se va a mover el contenido al hacer clic en la guía de la barra de desplazamiento horizontal. El valor predeterminado es 20.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se establece la propiedad `hPageScrollSize` para desplazarse 100 píxeles cuando el usuario haga clic en la guía de deslizamiento de la barra de desplazamiento horizontal.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Desplazar 100 píxeles cuando se haga clic en la barra horizontal.
my_sp.hPageScrollSize = 100;
```

ScrollPane.hPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hPosition

Descripción

Propiedad; orienta el contenido del panel de desplazamiento en píxeles y ajusta de forma proporcional el cuadro de desplazamiento (deslizador) horizontal. La posición 0 se encuentra en el extremo izquierdo de la guía de desplazamiento, lo que provoca que el borde izquierdo del contenido del panel de desplazamiento sea visible en el panel de desplazamiento.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se crea un detector para controlar el evento scroll y mostrar las posiciones de desplazamiento horizontal (`hPosition`) y vertical (`vPosition`) cuando el usuario haga clic en la barra de desplazamiento.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Desplazar 100 píxeles cuando se haga clic en la barra horizontal.
my_sp.hPageScrollSize = 100;

// Crear un objeto detector.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object) {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
}
// Añadir detector.
my_sp.addEventListener("scroll", spListener);
```

ScrollPane.hScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre (`"on"`), nunca (`"off"`) o aparece automáticamente en función del tamaño de la imagen (`"auto"`). El valor predeterminado es `"auto"`.

Ejemplo

En el siguiente ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se establece `hScrollPolicy` en `off` para evitar que aparezca una barra de desplazamiento horizontal, y se carga con una imagen.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);
my_sp.hScrollPolicy = "off";

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.progress

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object) {
  // ...
};
scrollPaneInstance.addEventListener("progress", listenerObject);
```

Sintaxis 2:

```
on (progress) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. El evento de progreso no siempre se difunde; el evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede especialmente cuando el contenido cargado es un archivo local. La aplicación activa el evento `progress` cuando la carga del contenido se inicia al establecer el valor de la propiedad `contentPath`.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*`scrollPaneInstance`*) distribuye un evento (en este caso, `progress`) y éste se controla mediante una función, también denominada *controlador*, en un objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de ScrollPane. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `mySPComponent` del componente ScrollPane, envía "_level0.mySPComponent" al panel Salida:

```
on (progress) {
  trace(this);
}
```

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp` y se define un objeto detector denominado `spListener` con un controlador de eventos `progress`. El controlador de eventos llama a las funciones `getBytesLoaded()` y `getBytesTotal()` para mostrar el progreso de la carga en el panel Salida.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var spListener:Object = new Object();
spListener.progress = function(evt_obj:Object):Void {
 trace("Loading " + my_sp.contentPath);
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded");
};
my_sp.addEventListener("progress", spListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.refreshPane()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.refreshPane()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; actualiza el panel de desplazamiento después de cargar el contenido. Este método vuelve a cargar el contenido pero no restablece la barra de desplazamiento. Utilice este método si, por ejemplo, ha cargado un formulario en un panel de desplazamiento y se ha modificado una propiedad de entrada (por ejemplo, un campo de texto) con ActionScript. En tal caso, llame a `refreshPane()` para volver a cargar el mismo formulario con los nuevos valores para las propiedades de entrada.

Ejemplo

En este ejemplo se crea un botón Refresh y una instancia de ScrollPane denominada `my_sp`. Se carga el componente ScrollPane con una imagen y se crea un detector para un evento click en el botón. Cuando se produce un evento click, el ejemplo llama a la función `refreshPane()`, que vuelve a cargar el contenido del panel de desplazamiento.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.controls.Button, "my_button", 10,
  {label:"Refresh"});
this.createClassObject(mx.containers.ScrollPane, "my_sp", 20);
my_sp.move(0, 30);
my_sp.setSize(360, 280);
```

```
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 my_sp.refreshPane();
}
my_button.addEventListener("click", buttonListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object):Void {
  // ...
};
scrollPaneInstance.addEventListener("scroll", listenerObject);
```

Sintaxis 2:

```
on (scroll) {
  // ...
}
```

Objeto de evento

Además de las propiedades estándar del objeto de evento, existen dos propiedades adicionales definidas para el evento `scroll`: una propiedad `type` cuyo valor es `"scroll"` y una propiedad `direction` cuyo valor puede ser `"vertical"` o `"horizontal"`.

Además de las propiedades estándar del objeto de evento, existen dos propiedades adicionales definidas para el evento `ProgressBar.progress`: `current` (el valor cargado igual a `total`) y `total` (el valor total).

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario hace clic en los botones de la barra de desplazamiento, el cuadro de desplazamiento (deslizador) o la guía de desplazamiento. A diferencia de otros eventos, el evento `scroll` se difunde cuando un usuario presiona el botón del ratón en la barra de desplazamiento y continúa difundiéndose hasta que se suelta el botón.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*`scrollPaneInstance`*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de ScrollPane. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `my_sp`, envía "_level0.my_sp" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se crea un detector para el evento scroll. Cuando se produce un evento scroll, el ejemplo muestra las posiciones de desplazamiento horizontal (`hPosition`) y vertical (`vPosition`) en el panel Salida.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Desplazar 100 píxeles cuando se haga clic en la barra horizontal.
my_sp.hPageScrollSize = 100;

// Crear un objeto detector.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object):Void {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Añadir detector.
my_sp.addEventListener("scroll", spListener);
```

Véase también

`EventDispatcher.addEventListener()`

ScrollPane.scrollDrag

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollPaneInstance``.scrollDrag`

Descripción

Propiedad; valor booleano que indica si se produce el desplazamiento (`true`) o no (`false`) cuando un usuario arrastra el contenido en el panel de desplazamiento. El valor predeterminado es `false`.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se establece la propiedad `scrollDrag` en `true`, lo que permite al usuario desplazarse arrastrando la imagen por el panel de desplazamiento.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Activar desplazamiento arrastrando panel de desplazamiento.
my_sp.scrollDrag = true;
```

ScrollPane.vLineScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollPaneInstance``.vLineScrollSize`

Descripción

Propiedad; número de píxeles que se va a mover el contenido en el área de visualización cuando el usuario haga clic en una flecha de desplazamiento de la barra de desplazamiento vertical. El valor predeterminado es 5.

Ejemplo

En el siguiente ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se establece la propiedad `vLineScrollSize` para desplazarse 20 píxeles cuando el usuario haga clic en una flecha de la barra de desplazamiento vertical.

Primero, arrastre el componente ScrollPane desde el panel Componentes al panel del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Desplazar 20 píxeles cuando se haga clic en las flechas de la barra
  vertical.
my_sp.vLineScrollSize = 20;
```

ScrollPane.vPageScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollPaneInstance``.vPageScrollSize`

Descripción

Propiedad; número de píxeles que se va a mover el contenido en el área de visualización cuando el usuario haga clic en la guía de una barra de desplazamiento vertical. El valor predeterminado es 20.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se establece la propiedad `vPageScrollSize` para desplazarse 30 píxeles cuando el usuario haga clic en la guía de deslizamiento de la barra de desplazamiento vertical.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Desplazar 30 píxeles cuando se haga clic en la barra vertical.
my_sp.vPageScrollSize = 30;
```

ScrollPane.vPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollPaneInstance``.vPosition`

Descripción

Propiedad; orienta el contenido del panel de desplazamiento en píxeles y ajusta de forma proporcional el cuadro de desplazamiento (deslizador) vertical. La posición 0 se encuentra en el extremo superior de la guía de desplazamiento, lo que provoca que el borde superior del contenido del panel de desplazamiento sea visible en el panel de desplazamiento. El valor predeterminado es 0.

Ejemplo

En este ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se carga con una imagen y se crea un detector para controlar el evento scroll y mostrar las posiciones de desplazamiento horizontal (`hPosition`) y vertical (`vPosition`) cuando el usuario haga clic en la barra de desplazamiento.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Desplazar 100 píxeles cuando se haga clic en la barra horizontal.
my_sp.hPageScrollSize = 100;

// Crear un objeto detector.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object):Void {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Añadir detector.
my_sp.addEventListener("scroll", spListener);
```

ScrollPane.vScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.vScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre (`"on"`), nunca (`"off"`) o aparece automáticamente en función del tamaño de la imagen (`"auto"`). El valor predeterminado es `"auto"`.

Ejemplo

En el siguiente ejemplo se crea una instancia de ScrollPane denominada `my_sp`, se establece `vScrollPolicy` en `off` para evitar que aparezca una barra de desplazamiento vertical, y se carga el componente ScrollPane con una imagen.

Primero, arrastre el componente ScrollPane desde el panel Componentes a la biblioteca del documento actual y, a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente ScrollPane en la biblioteca
*/
import mx.containers.ScrollPane;

this.createClassObject(ScrollPane, "my_sp", 30);
my_sp.setSize(360, 280);
my_sp.vScrollPolicy = "off";

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

CAPÍTULO 41

Clase SimpleButton

41

Herencia MovieClip > Clase UIObject > Clase UIComponent > SimpleButton

Nombre de clase de ActionScript mx.controls.SimpleButton

Las propiedades de la clase SimpleButton permiten controlar lo siguiente en tiempo de ejecución:

- Si el botón tiene el aspecto de un botón de comando predeterminado
- Si el botón actúa como un botón de comando o como un conmutador
- Si el botón está seleccionado

Resumen de métodos de la clase SimpleButton

No hay métodos exclusivos de la clase SimpleButton.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase SimpleButton de la clase UIObject. Al llamar a estos métodos desde la clase SimpleButton, debe utilizarse la forma *buttonInstance.methodName*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.

Método	Descripción
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase SimpleButton de la clase UIComponent. Al llamar a estos métodos desde el objeto SimpleButton, debe utilizarse la forma *`buttonInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase SimpleButton

En la tabla siguiente se enumeran las propiedades de la clase SimpleButton.

Propiedad	Descripción
`SimpleButton.emphasized`	Indica si el botón tiene el aspecto de un botón de comando predeterminado.
`SimpleButton.emphasizedStyleDeclaration`	Declaración de estilos cuando la propiedad `emphasized` está definida en `true`.
`SimpleButton.selected`	Valor booleano que indica si el botón está seleccionado (`true`) o no (`false`). El valor predeterminado es `false`.
`SimpleButton.toggle`	Valor booleano que indica si el botón se comporta como un conmutador (`true`) o no (`false`). El valor predeterminado es `false`.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase SimpleButton de la clase UIObject. Al acceder a estas propiedades desde el objeto SimpleButton, debe utilizarse la forma *buttonInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
UIObject.height	Altura del objeto, expresada en píxeles. Sólo lectura.
UIObject.left	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase SimpleButton de la clase UIComponent. Al acceder a estas propiedades desde el objeto SimpleButton, debe utilizarse la forma *buttonInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase SimpleButton

En la tabla siguiente se muestra el evento de la clase SimpleButton.

Evento	Descripción
SimpleButton.click	Se difunde cuando se hace clic en un botón.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase SimpleButton de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase SimpleButton de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

SimpleButton.click

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object){
  // ...
};
buttonInstance.addEventListener("click", listenerObject);
```

Sintaxis 2:

```
on (click) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) en un botón o cuando éste está seleccionado y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*buttonInstance*) distribuye un evento (en este caso, `click`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se produce el evento. Cuando se produce el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama a `addEventListener()` (véase `EventDispatcher.addEventListener()`) en la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente Button. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia del componente Button `myButtonComponent`, envía "_level0.myButtonComponent" al panel Salida:

```
on (click) {
  trace(this);
}
```

El comportamiento de `this` es diferente cuando se utiliza dentro de un controlador `on()` asociado con un símbolo de botón normal de Flash; en esta instancia, `this` hace referencia a la línea de tiempo que contiene el botón. Por ejemplo, el código siguiente, asociado con la instancia `myButton` del símbolo de botón, envía "_level0" al panel Salida:

```
on (release) {
  trace(this);
}
```

NOTA El objeto incorporado Button de ActionScript no tiene eventos `click`; el evento más próximo es `release`.

Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se hace clic en un botón denominado `buttonInstance`. La primera línea especifica que el botón actúe como un conmutador. La segunda línea crea un objeto detector denominado `form`. La tercera línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una sentencia `trace()` que utiliza el objeto de evento que se pasa automáticamente a la función (en este ejemplo `eventObj`) para generar un mensaje. La propiedad `target` de un objeto de evento es el componente que ha generado el evento (en este ejemplo `buttonInstance`). Se accede a la propiedad `SimpleButton.selected` desde la propiedad `target` del objeto de evento. La última línea llama a `addEventListener()` desde `buttonInstance` y pasa como parámetros el evento `click` y el objeto detector `form`.

```
buttonInstance.toggle = true;
var form:Object = new Object();
form.click = function(eventObj:Object) {
  trace("The selected property has changed to " +
  eventObj.target.selected);
};
buttonInstance.addEventListener("click", form);
```

El código siguiente envía también un mensaje al panel Salida cuando se hace clic en `buttonInstance`. El controlador `on()` debe asociarse directamente con `buttonInstance`.

```
on (click) {
  trace("button component was clicked");
}
```

SimpleButton.emphasized

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

buttonInstance.emphasized

Descripción

Propiedad; indica si el botón está resaltado (`true`) o no (`false`). El estado resaltado equivale al aspecto de un botón de comando predeterminado. En general, utilice la propiedad `FocusManager.defaultPushButton` en lugar de definir directamente la propiedad `emphasized`. El valor predeterminado es `false`.

Si no está utilizando `FocusManager.defaultPushButton`, es posible que sólo desee definir un botón en el estado emphasized, o utilizar el estado emphasized para cambiar el texto de un color a otro. En el siguiente ejemplo se define la propiedad `emphasized` de la instancia del botón `myButton`:

```
_global.styles.foo = new CSSStyleDeclaration();
_global.styles.foo.color = 0xFF0000;
SimpleButton.emphasizedStyleDeclaration = "neutralStyle";
myButton.emphasized = true;
```

Véase también

`SimpleButton.emphasizedStyleDeclaration`

SimpleButton.emphasizedStyleDeclaration

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

buttonInstance.emphasizedStyleDeclaraion

Descripción

Propiedad (estática); cadena que indica la declaración de estilo que da formato a un botón cuando la propiedad `emphasized` se define en `true`.

La propiedad `emphasizedStyleDeclaration` es una propiedad estática de la clase SimpleButton. Por tanto, debe acceder a ella directamente desde SimpleButton y no desde *`buttonInstance`*, como en el ejemplo siguiente:

```
SimpleButton.emphasizedStyleDeclaration = "3dEmphStyle";
```

Véase también

SimpleButton.emphasized

SimpleButton.selected

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`buttonInstance``.selected`

Descripción

Propiedad; valor booleano que indica si el botón está seleccionado (`true`) o no (`false`). El valor predeterminado es `false`.

SimpleButton.toggle

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`buttonInstance``.toggle`

Descripción

Propiedad; valor booleano que indica si el botón actúa como un conmutador (`true`) o no (`false`). El valor predeterminado es `false`.

Si un botón actúa como un conmutador, permanece presionado hasta que se hace clic nuevamente en él.

CAPÍTULO 42

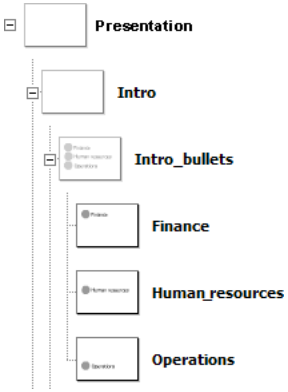
Clase Slide (sólo en Flash Professional)

42

La clase Slide corresponde a un nodo de la presentación jerárquica de diapositivas. En Flash Professional 8 puede crear presentaciones de diapositivas mediante el panel Contorno de pantalla. Para ver información general sobre cómo trabajar con pantallas, consulte Capítulo 14, "Trabajo con pantallas (sólo para Flash Professional)" en *Utilización de Flash*.

La clase Slide amplía la clase Screen (véase "Clase Screen (sólo en Flash Professional)" en la página 1109) y ofrece capacidades de navegación y secuenciación incorporadas entre las diapositivas, además de la capacidad de asociar fácilmente transiciones entre diapositivas mediante comportamientos. Las diapositivas almacenan información sobre su "estado", de forma que cuando el usuario avanza a la diapositiva adyacente, la diapositiva anterior se oculta.

Tenga en cuenta que sólo puede desplazarse por ("detenerse en") las diapositivas que no contienen diapositivas secundarias (diapositivas "hoja"). Por ejemplo, la ilustración siguiente muestra el contenido del panel Contorno de pantalla de un ejemplo de presentación de diapositivas.



Cuando se inicie dicha presentación, se "detendrá" de forma predeterminada en la diapositiva llamada Finance, que es la primera diapositiva de la presentación que no contiene diapositivas secundarias.

Asimismo, tenga en cuenta que las diapositivas secundarias "heredan" el aspecto visual (gráficos y el resto del contenido) de las diapositivas principales. Por ejemplo, en la ilustración anterior, además del contenido de la diapositiva Finance, el usuario puede ver todo el contenido de las diapositivas Intro y Presentation.

NOTA

La clase Slide hereda de la Clase Loader, que permite cargar fácilmente archivos SWF o JPEG externos en una diapositiva determinada. Esto permite dividir en módulos las presentaciones de diapositivas y reducir el tiempo de descarga inicial. Para más información, consulte "Carga de contenido externo en pantallas (sólo en Flash Professional)" en la página 1110.

Utilización de la clase Slide (sólo en Flash Professional)

Los métodos y las propiedades de la clase Slide se utilizan para controlar las presentaciones de diapositivas que se crean mediante el panel Contorno de pantalla de una presentación de diapositivas de Flash, para obtener información sobre la presentación de diapositivas (por ejemplo, para determinar el número de diapositivas secundarias que contiene una diapositiva principal) o para desplazarse entre las diapositivas de una presentación (por ejemplo, para crear botones de "Diapositiva siguiente" y "Diapositiva anterior").

También es posible utilizar los comportamientos incorporados disponibles en el panel Comportamientos para controlar las presentaciones de diapositivas. Para más información, consulte "Adición de control a las pantallas mediante comportamientos (sólo en Flash Professional)" en *Utilización de Flash*.

Parámetros de Slide

A continuación se indican los parámetros de edición que se pueden definir para cada diapositiva en el inspector de propiedades o en el inspector de componentes:

autoKeyNav determina si la diapositiva responde a la navegación predeterminada mediante el teclado y, en tal caso, también determina el modo en que lo hace. Para más información, consulte `Slide.autoKeyNav`.

autoload indica si el contenido especificado por el parámetro contentPath debe cargarse automáticamente (`true`) o si debe esperar hasta que se llame al método `Loader.load()` (`false`). El valor predeterminado es `true`.

contentPath especifica el contenido de la diapositiva. Este valor puede ser el identificador de vinculación de un clip de película o la URL absoluta o relativa de un archivo SWF o JPEG que se carga en la diapositiva. De forma predeterminada, el contenido cargado se recorta para que quepa en la diapositiva.

overlayChildren especifica si las diapositivas secundarias de la diapositiva siguen viéndose al desplazarse de una diapositiva secundaria a la siguiente (`true`) o no (`false`).

playHidden especifica si la diapositiva continúa reproduciéndose cuando está oculta (`true`) o no (`false`).

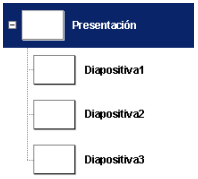
Utilización de la clase Slide para crear una presentación de diapositivas

Los métodos y las propiedades de la clase Slide se utilizan para controlar las presentaciones de diapositivas que se crean en el panel Contorno de pantalla de una presentación de diapositivas de Flash, en el entorno de edición de Flash. El panel Comportamientos también contiene varios comportamientos para crear desplazamientos entre diapositivas. En este ejemplo, se escribe código ActionScript para crear botones Siguiente y Anterior para una presentación de diapositivas.

Para crear una presentación de diapositivas con navegación:

1. En Flash, seleccione Archivo > Nuevo.
2. En la ficha General, seleccione Presentación de Flash.
3. En el panel Contorno de pantalla, haga clic en el botón Insertar pantalla (+) dos veces para crear dos diapositivas debajo de la diapositiva de presentación.

   El panel Contorno de pantalla tiene el aspecto siguiente:



4. Seleccione diapositiva1 en el panel Contorno de pantalla y, utilizando la herramienta Texto, añada un campo de texto en el que se lea **Ésta es la diapositiva 1**.
5. Repita el paso anterior para diapositiva2 y diapositiva3 y cree campos de texto en cada diapositiva en los que se lea **Ésta es la diapositiva 2** y **Ésta es la diapositiva 3**, respectivamente.
6. Seleccione la diapositiva Presentación y abra el panel Componentes.

7. Arrastre un componente Button desde el panel Componentes a la parte inferior del escenario.
8. En el inspector de propiedades, escriba **Diapositiva siguiente** para la propiedad Label del componente Button.
9. En el panel Acciones, especifique el siguiente código:

```
on(click) {
  _parent.currentSlide.gotoNextSlide();
}
```

10. Pruebe el archivo SWF (Control > Probar película) y pase a la diapositiva siguiente haciendo clic en el botón Diapositiva siguiente.

Clase Slide (API) (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > Componente Loader > Clase Screen (sólo en Flash Professional) > Slide

Nombre de clase de ActionScript mx.screens.Slide

Los métodos, propiedades y eventos de la clase Slide permiten administrar y manipular diapositivas.

Resumen de métodos de la clase Slide

En la tabla siguiente se enumeran los métodos de la clase Slide:

Método	Descripción
`Slide.getChildSlide()`	Devuelve la diapositiva secundaria especificada.
`Slide.gotoFirstSlide()`	Se desplaza a la primera diapositiva hoja de la jerarquía de subdiapositivas de la diapositiva.
`Slide.gotoLastSlide()`	Se desplaza a la última diapositiva hoja de la jerarquía de subdiapositivas de la diapositiva.
`Slide.gotoNextSlide()`	Se desplaza a la diapositiva siguiente.
`Slide.gotoPreviousSlide()`	Se desplaza a la diapositiva anterior.
`Slide.gotoSlide()`	Se desplaza a una diapositiva especificada.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Slide de la clase UIObject. Al llamar a estos métodos desde el objeto Slide, debe utilizarse la forma *SlideInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Slide de la clase UIComponent. Al llamar a estos métodos desde el objeto Slide, debe utilizarse la forma *SlideInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Métodos heredados de la clase Loader

En la tabla siguiente se enumeran los métodos que hereda la clase Slide de la clase Loader. Al llamar a este método desde el objeto Slide, debe utilizarse la forma *`SlideInstance.methodName`*.

Método	Descripción
`Loader.load()`	Carga el contenido especificado en la propiedad `contentPath`.

Métodos heredados de la clase Screen

En la tabla siguiente se enumeran los métodos que hereda la clase Slide de la clase Screen. Al llamar a este método desde el objeto Slide, debe utilizarse la forma *`SlideInstance.methodName`*.

Método	Descripción
`Screen.getChildScreen()`	Devuelve la pantalla secundaria de esta pantalla en un índice concreto.

Resumen de propiedades de la clase Slide

En la tabla siguiente se enumeran las propiedades de la clase Slide:

Propiedad	Descripción
`Slide.autoKeyNav`	Determina si se utilizan las funciones predeterminadas del teclado para desplazarse a la diapositiva siguiente o anterior.
`Slide.currentChildSlide`	Sólo lectura; devuelve la diapositiva secundaria inmediata de la diapositiva especificada que contiene la diapositiva que se encuentra activa.
`Slide.currentFocusedSlide`	Sólo lectura; devuelve la "última diapositiva hoja" (la más alejada de la raíz del árbol de diapositivas) que contiene la selección actual global.
`Slide.currentSlide`	Sólo lectura; devuelve la diapositiva que se encuentra activa.
`Slide.defaultKeydownHandler`	Función callback que sustituye la navegación predeterminada mediante el teclado (teclas de flecha izquierda y derecha).
`Slide.firstSlide`	Sólo lectura; devuelve la primera diapositiva secundaria de la diapositiva que no tiene elementos secundarios.

Propiedad	Descripción
`Slide.indexInParentSlide`	Sólo lectura; devuelve el índice de la diapositiva (basado en cero) en la lista de subdiapositivas de su diapositiva principal.
`Slide.lastSlide`	Sólo lectura; devuelve la última diapositiva secundaria de la diapositiva que no tiene elementos secundarios.
`Slide.nextSlide`	Sólo lectura; devuelve la diapositiva a la que se llegaría si se llama a `mySlide.gotoNextSlide()`, pero sin que se desplace realmente a dicha diapositiva.
`Slide.numChildSlides`	Sólo lectura; devuelve el número de diapositivas secundarias que contiene la diapositiva.
`Slide.overlayChildren`	Determina si las diapositivas secundarias de la diapositiva están visibles cuando se pasa de una diapositiva secundaria a la siguiente.
`Slide.parentIsSlide`	Sólo lectura; devuelve un valor booleano que indica si el objeto principal de la diapositiva también es una diapositiva (`true`) o no (`false`).
`Slide.parentSlide`	Sólo lectura; diapositiva que contiene la diapositiva actual. Puede ser `null` para la diapositiva raíz.
`Slide.playHidden`	Determina si la diapositiva continúa reproduciéndose mientras está oculta.
`Slide.previousSlide`	Sólo lectura; devuelve la diapositiva a la que se llegaría si se llama a `mySlide.gotoPreviousSlide()`, pero sin que se desplace realmente a dicha diapositiva.
`Slide.rootSlide`	Sólo lectura; devuelve la raíz del árbol de diapositivas que contiene la diapositiva.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Slide de la clase UIObject. Al acceder a estas propiedades desde el objeto Slide, debe utilizarse la forma *`SlideInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.

Propiedad	Descripción
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Slide de la clase UIComponent. Al acceder a estas propiedades desde el objeto Slide, debe utilizarse la forma *`SlideInstance.propertyName`*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase Loader

En la tabla siguiente se enumeran las propiedades que hereda la clase Slide de la clase Loader. Al acceder a estas propiedades desde el objeto Slide, debe utilizarse la forma *`SlideInstance.propertyName`*.

Propiedad	Descripción
`Loader.autoLoad`	Valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso llamar a `Loader.load()` (`false`).
`Loader.bytesLoaded`	Propiedad de sólo lectura que indica el número de bytes que se han cargado.
`Loader.bytesTotal`	Propiedad de sólo lectura que indica el número total de bytes del contenido.
`Loader.content`	Referencia al contenido del componente Loader. Es una propiedad de sólo lectura.
`Loader.contentPath`	Cadena que indica la URL del contenido que va a cargarse.
`Loader.percentLoaded`	Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.
`Loader.scaleContent`	Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader (`true`) o si el componente Loader se redimensiona para adaptarse al contenido (`false`).

Propiedades heredadas de la clase Screen

En la tabla siguiente se enumeran las propiedades que hereda la clase Slide de la clase Screen. Al acceder a estas propiedades desde el objeto Slide, debe utilizarse la forma *`SlideInstance.propertyName`*.

Propiedad	Descripción
`Screen.currentFocusedScreen`	Sólo lectura; devuelve la pantalla que contiene la selección actual global.
`Screen.indexInParent`	Sólo lectura; devuelve el índice de la pantalla (basado en cero) en la lista de pantallas secundarias de su pantalla principal.
`Screen.numChildScreens`	Sólo lectura; devuelve el número de pantallas secundarias que contiene la pantalla.

Propiedad	Descripción
`Screen.parentIsScreen`	Sólo lectura; devuelve un valor booleano (`true` o `false`) que indica si el objeto principal de la pantalla también es una pantalla.
`Screen.rootScreen`	Sólo lectura; devuelve la pantalla raíz del árbol o subárbol que contiene la pantalla.

Resumen de eventos de la clase Slide

En la tabla siguiente se enumeran los eventos de la clase Slide.

Evento	Descripción
`Slide.hideChild`	Se difunde cada vez que un elemento secundario de una diapositiva pasa de ser visible a invisible.
`Slide.revealChild`	Se difunde cada vez que una diapositiva secundaria de un objeto de diapositiva pasa de ser invisible a visible.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Slide de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Slide de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase Loader

En la tabla siguiente se enumeran los eventos que hereda la clase Slide de la clase Loader.

Evento	Descripción
`Loader.complete`	Se activa cuando el contenido termina de cargarse.
`Loader.progress`	Se activa mientras se carga el contenido.

Eventos heredados de la clase Screen

En la tabla siguiente se enumeran los eventos que hereda la clase Slide de la clase Screen.

Evento	Descripción
`Screen.allTransitionsInDone`	Se difunde cuando todas las transiciones de entrada aplicadas a una pantalla han finalizado.
`Screen.allTransitionsOutDone`	Se difunde cuando todas las transiciones de salida aplicadas a una pantalla han finalizado.
`Screen.mouseDown`	Se difunde cuando se ha presionado el botón del ratón sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
`Screen.mouseDownSomewhere`	Se difunde cuando se ha presionado el botón del ratón en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.
`Screen.mouseMove`	Se difunde cuando el ratón se mueve mientras se encuentra sobre una pantalla.
`Screen.mouseOut`	Se difunde cuando el ratón se mueve del interior al exterior de la pantalla.

Evento	Descripción
`Screen.mouseOver`	Se difunde cuando el ratón se mueve del exterior al interior de la pantalla.
`Screen.mouseUp`	Se difunde cuando el botón del ratón se ha soltado sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.
`Screen.mouseUpSomewhere`	Se difunde cuando el botón del ratón se ha soltado en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.

Slide.autoKeyNav

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.autoKeyNav

Descripción

Propiedad; determina si la diapositiva utiliza o no las funciones predeterminadas del teclado para desplazarse a la diapositiva siguiente o anterior cuando se ha seleccionado *mySlide*. Esta propiedad acepta los valores de cadena `"true"`, `"false"` y `"inherit"`. También puede sustituir el comportamiento de las funciones predeterminadas del teclado mediante la propiedad `Slide.defaultKeydownHandler`.

Si se ha establecido el valor de esta propiedad en `"true"` y se presiona la tecla de flecha derecha (`Key.RIGHT`) o la barra espaciadora (`Key.SPACE`) cuando se ha seleccionado *mySlide*, se avanzará hasta la diapositiva siguiente. En cambio, si se presiona la tecla de flecha izquierda (`Key.Left`) se desplazará a la diapositiva anterior.

Si el valor de esta propiedad es `"false"`, las funciones predeterminadas del teclado no tendrán efecto alguno si se ha seleccionado *mySlide*.

Si el valor de esta propiedad es `"inherit"`, *mySlide* comprueba la propiedad `autoKeyNav` de su diapositiva principal. Si el valor de ésta también es `"inherit"`, Flash busca la cadena de herencia de diapositivas hasta que encuentra una diapositiva principal cuya propiedad `autoKeyNav` tenga el valor `"true"` o `"false"`.

Si *mySlide* no tiene ninguna diapositiva principal (es decir, si el valor de `(mySlide.parentIsSlide == false)` es `true`) se comportará como si `autoKeyNav` se hubiera establecido en `"true"`.

Ejemplo

En este ejemplo se desactiva la navegación automática mediante el teclado para la diapositiva denominada `loginSlide`.

```
_root.Presentation.loginSlide.autoKeyNav = "false";
```

Véase también

`Slide.defaultKeydownHandler`

Slide.currentChildSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide`.currentChildSlide`

Descripción

Propiedad (sólo lectura); devuelve la diapositiva secundaria inmediata de *mySlide* que contiene la diapositiva que se encuentra activa; devuelve `null` si no se encuentra seleccionada ninguna de las diapositivas secundarias de *mySlide*.

Ejemplo

Observe el contorno de pantalla siguiente:

```
Presentation
  Slide_1
    Bullet1_1
      SubBullet1_1_1
    Bullet1_2
      SubBullet1_2_1
  Slide_2
```

Si se toma `SubBullet1_1_1` como la diapositiva actual, todas las sentencias siguientes serán verdaderas:

```
Presentation.currentChildSlide == Slide_1;
Slide_1.currentChildSlide == Bullet_1_1;
SubBullet_1_1_1.currentChildSlide == null;
Slide_2.currentChildSlide == null;
```

Véase también

`Slide.currentSlide`

Slide.currentFocusedSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
mx.screens.Slide.currentFocusedSlide
```

Descripción

Propiedad (sólo lectura); devuelve la "última diapositiva hoja" (la más alejada de la raíz del árbol de diapositivas) que contiene la selección actual global. La selección puede recaer en la propia diapositiva o en un clip de película, objeto de texto o componente de dicha diapositiva. El método devuelve `null` si no se ha seleccionado ningún elemento.

Ejemplo

```
var focusedSlide = mx.screens.Slide.currentFocusedSlide;
```

Slide.currentSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
mySlide.currentSlide
```

Descripción

Propiedad (sólo lectura); devuelve la diapositiva que se encuentra activa. Se trata siempre de una diapositiva "hoja", es decir, una diapositiva que no contiene diapositivas secundarias.

Ejemplo

El código siguiente, asociado con un botón de la diapositiva de presentación raíz, avanza por la presentación de diapositivas hasta la diapositiva siguiente cada vez que se hace clic en el botón.

```
// Asociado con la instancia de botón contenida en la diapositiva de
  presentación:
on(press) {
  _parent.currentSlide.gotoNextSlide();
}
```

Véase también

`Slide.gotoNextSlide()`

Slide.defaultKeydownHandler

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
mySlide.defaultKeyDownHandler = function (eventObj) {
  // El código se escribe aquí.
}
```

Parámetros

`eventObj` Objeto de evento con las propiedades siguientes:

- `type` Cadena que indica el tipo de evento. Los valores posibles son `"keyUp"` y `"keyDown"`.
- `ascii` Número entero que representa el valor ASCII de la última tecla presionada; corresponde al valor devuelto por `Key.getAscii()`.
- `code` Número entero que representa el código de tecla de la última tecla presionada; corresponde al valor devuelto por `Key.getCode()`.
- `shiftKey` Valor booleano que indica si la tecla Mayús está presionada actualmente `true`) o no (`false`).

- `ctrlKey` Valor booleano que indica si la tecla Control está presionada actualmente (`true`) o no (`false`).

Valor devuelto

Ninguno.

Descripción

Función callback; permite sustituir la navegación predeterminada mediante el teclado por un controlador de teclado personalizado creado por el usuario. Por ejemplo, en lugar de utilizar las teclas de flecha izquierda y derecha para desplazarse a las diapositivas anterior y siguiente de una presentación respectivamente, puede hacer que las teclas de flecha arriba y abajo lleven a cabo dichas funciones. Para más información sobre el comportamiento de las funciones predeterminadas del teclado, consulte `Slide.autoKeyNav`.

Ejemplo

En este ejemplo se modifican las funciones predeterminadas del teclado para las diapositivas secundarias de la diapositiva con la que se encuentra asociado el controlador `on(load)`. Este controlador utiliza las teclas de flecha arriba y abajo para la navegación, en lugar de las teclas de flecha izquierda y derecha.

```
on (load) {
  this.defaultKeyDownHandler = function(eventObj:Object) {
    switch (eventObj.code) {
    case Key.DOWN :
      this.currentSlide.gotoNextSlide();
      break;
    case Key.UP :
      this.currentSlide.gotoPreviousSlide();
      break;
    default :
      break;
    }
  };
}
```

Véase también

`Slide.autoKeyNav`

Slide.firstSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.firstSlide

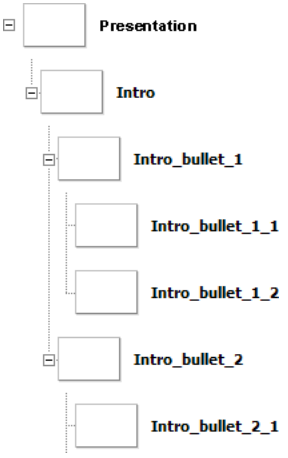
Descripción

Propiedad (sólo lectura); devuelve la primera diapositiva secundaria de *mySlide* que no tenga diapositivas secundarias.

Ejemplo

En la jerarquía de diapositivas que se muestra a continuación, las dos sentencias siguientes son verdaderas:

```
Presentation.Intro.firstSlide == Intro_bullet_1_1;
Presentation.Intro_bullet_1.firstSlide == Intro_bullet_1_1;
```



Slide.getChildSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.getChildSlide(*childIndex*)

Parámetros

childIndex Índice basado en cero de la diapositiva secundaria que se devuelve.

Valor devuelto

Un objeto de diapositiva.

Descripción

Método; devuelve la diapositiva secundaria de *mySlide* cuyo índice es *childIndex*. Este método resulta útil para repetir un conjunto de diapositivas secundarias cuyos índices son conocidos.

Ejemplo

Como resultado del código siguiente, el panel Salida muestra los nombres de todas las diapositivas secundarias de la diapositiva de presentación raíz.

```
var numSlides = _root.Presentation.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
  var childSlide = _root.Presentation.getChildSlide(slideIndex);
  trace(childSlide._name);
}
```

Véase también

Slide.numChildSlides

Slide.gotoFirstSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.gotoFirstSlide()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; se desplaza hasta la primera diapositiva hoja del árbol de las diapositivas secundarias que se encuentran debajo de *mySlide*. Este método se omite cuando se le llama desde un controlador de eventos `on(hide)` o `on(reveal)` de una diapositiva, en caso de que dicho evento se haya creado al desplazarse por las diapositivas.

Para ir a la primera diapositiva de una presentación, llame a *mySlide*.rootSlide.gotoFirstSlide(). Para más información sobre `rootSlide`, consulte `Slide.revealChild`.

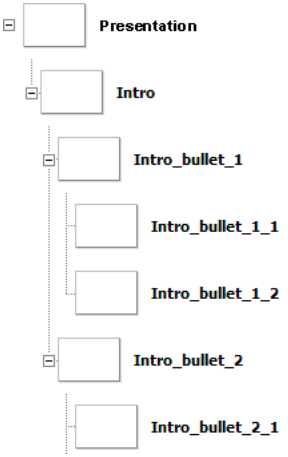
Ejemplo

En la jerarquía de diapositivas que aparece a continuación, todas las llamadas al método se desplazan hasta la diapositiva denominada `Intro_bullet_1_1`:

```
Presentation.gotoFirstSlide();
Presentation.Intro.gotoFirstSlide();
Presentation.Intro.Intro_bullet_1.gotoFirstSlide();
```

Esta llamada al método se desplaza hasta la diapositiva denominada `Intro_bullet_2_1`:

```
Presentation.Intro.Intro_bullet_2.gotoFirstSlide();
```



Véase también

`Slide.firstSlide`, `Slide.revealChild`

Slide.gotoLastSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.gotoLastSlide()`

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; se desplaza a la última diapositiva hoja del árbol de diapositivas secundarias debajo de *mySlide*. Este método se omite cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` de una diapositiva, en caso de que dicho evento sea resultado de otro desplazamiento entre diapositivas.

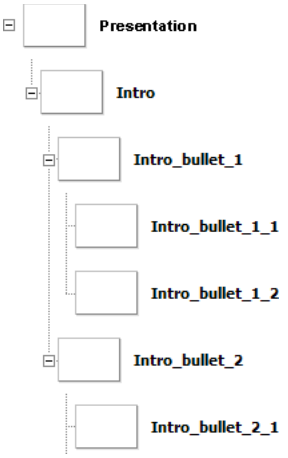
Ejemplo

En la jerarquía de diapositivas que aparece a continuación, las llamadas al método se desplazan hasta la diapositiva denominada `Intro_bullet_1_2`:

```
Presentation.Intro.gotoLastSlide();
Presentation.Intro.Intro_bullet_1.gotoLastSlide();
```

Esta llamada al método se desplaza hasta la diapositiva denominada `Intro_bullet_2_1`:

```
Presentation.gotoLastSlide();
Presentation.Intro.gotoLastSlide();
```



Véase también

`Slide.gotoSlide()`, `Slide.lastSlide`

Slide.gotoNextSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.gotoNextSlide()

Parámetros

Ninguno.

Valor devuelto

Valor booleano o `null`. El método devuelve `true` si se ha desplazado satisfactoriamente hasta la siguiente diapositiva; devuelve `false` si la presentación ya está en la última diapositiva cuando se invoca al método (es decir, si el valor de `currentSlide.nextSlide` es `null`). El método devuelve `null` si se invoca en una diapositiva que no contiene la diapositiva actual.

Descripción

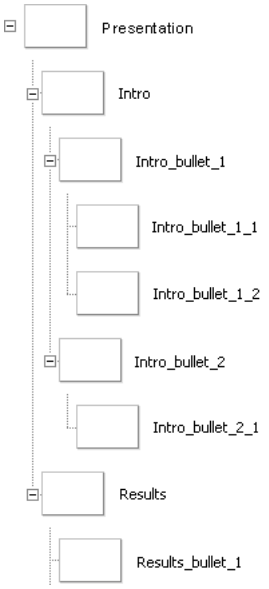
Método; se desplaza hasta la diapositiva siguiente de la presentación de diapositivas. Cuando se pasa de una diapositiva a la siguiente, la diapositiva de salida se oculta y aparece la diapositiva de entrada. Si las diapositivas de salida y entrada se encuentran en subárboles diferentes, todas las diapositivas ascendientes, empezando por la diapositiva de salida y hasta la diapositiva ascendiente común de ambas diapositivas, se ocultan y se recibe un evento `hide`. Inmediatamente después, aparecen todas las diapositivas ascendientes de la diapositiva de entrada, hasta la diapositiva ascendiente común de las diapositivas de entrada y de salida, y se recibe el evento `reveal`.

Normalmente, se llama a `gotoNextSlide()` en el nodo hoja que representa a la diapositiva actual. Si se llama a `someNode` en un nodo que no es hoja, `someNode.gotoNextSlide()` avanzará hasta el primer nodo hoja de la diapositiva o "sección" siguiente.

Este método no tiene ningún efecto cuando se invoca en una diapositiva que no contiene la diapositiva actual. El método tampoco tiene efecto cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` asociado con una diapositiva, en caso de que se haya invocado a dicho controlador debido a los desplazamientos entre diapositivas.

Ejemplo

Suponga que, en la jerarquía de diapositivas siguiente, la diapositiva denominada `Intro_bullet_1_1` es la diapositiva actual que aparece en pantalla (es decir, `_root.Presentation.currentSlide._name == Intro_bullet_1_1`).



En este caso, si se llama a `Intro_bullet_1_1.gotoNextSlide()` se desplazará hasta `Intro_bullet_1_2`, que es la diapositiva del mismo nivel que `Intro_bullet_1_1`.

Sin embargo, si se llama a `Intro_bullet_1.gotoNextSlide()` se desplazará hasta `Intro_bullet_2_1`, que es la primera diapositiva hoja de `Intro_bullet_2`, la cual, a su vez, es la siguiente diapositiva del mismo nivel de `Intro_bullet_1`. Del mismo modo, si se llama a `Intro.gotoNextSlide()` se desplazará hasta `Results_bullet_1`, que es la primera diapositiva hoja contenida en la diapositiva `Results`.

Asimismo, siendo la diapositiva actual `Intro_bullet_1_1`, si se llama a `Results.gotoNextSlide()`, éste no tendrá ningún efecto, puesto que la diapositiva `Results` no contiene la diapositiva actual (es decir, el valor de `Results.currentSlide` es `null`).

Véase también

`Slide.currentSlide`, `Slide.gotoPreviousSlide()`, `Slide.nextSlide`

Slide.gotoPreviousSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.gotoPreviousSlide()

Parámetros

Ninguno.

Valor devuelto

Valor booleano o `null`. El método devuelve `true` si se ha desplazado satisfactoriamente hasta la diapositiva anterior; devuelve `false` si la presentación ya está en la primera diapositiva cuando se invoca al método (es decir, si el valor de `currentSlide.nextSlide` es `null`). El método devuelve `null` si se invoca en una diapositiva que no contiene la diapositiva actual.

Descripción

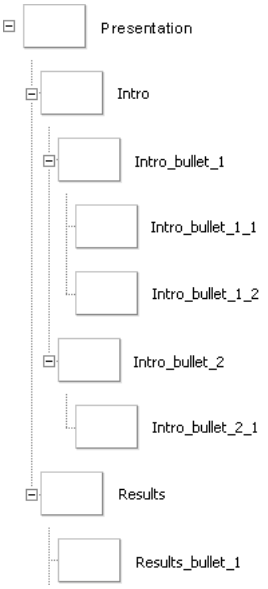
Método; se desplaza a la diapositiva anterior de la presentación de diapositivas. Al pasar de una diapositiva a la anterior, la diapositiva de salida queda oculta y aparece la diapositiva de entrada. Si las diapositivas de salida y entrada se encuentran en subárboles diferentes, todas las diapositivas ascendientes, empezando por la diapositiva de salida y hasta la diapositiva ascendiente común de ambas diapositivas, se ocultan y se recibe un evento `hide`. Inmediatamente después, aparecen todas las diapositivas ascendientes de la diapositiva de entrada, hasta la diapositiva ascendiente común de las diapositivas de entrada y salida, y se recibe el evento `reveal`.

Normalmente se llama al método `gotoPreviousSlide()` en el nodo hoja que representa la diapositiva actual. Si se llama a `someNode` en un nodo que no es hoja, `someNode.gotoPreviousSlide()` avanzará hasta el primer nodo hoja de la diapositiva o "sección" anterior.

Este método no tiene ningún efecto cuando se invoca en una diapositiva que no contiene la diapositiva actual. El método tampoco tiene efecto cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` asociado con una diapositiva, en caso de que se haya invocado a dicho controlador debido a los desplazamientos entre diapositivas.

Ejemplo

Suponga que, en la siguiente jerarquía de diapositivas, la diapositiva denominada `Intro_bullet_1_2` es la diapositiva actual que aparece en pantalla (es decir, `_root.Presentation.currentSlide._name == Intro_bullet_1_2`).



En este caso, si se llama a `Intro_bullet_1_2.gotoPreviousSlide()` se desplazará hasta `Intro_bullet_1_1`, que es la diapositiva anterior del mismo nivel que `Intro_bullet_1_2`.

Sin embargo, si se llama a `Intro_bullet_2.gotoPreviousSlide()` se desplazará hasta `Intro_bullet_1_1`, que es la primera diapositiva hoja de `Intro_bullet_1`, la cual, a su vez, es la anterior diapositiva del mismo nivel que `Intro_bullet_2`. De forma similar, si se llama a `Results.gotoPreviousSlide()` se desplazará hasta `Intro_bullet_1_1`, que es la primera diapositiva hoja contenida en la diapositiva `Intro`.

Asimismo, siendo la diapositiva actual `Intro_bullet_1_1`, si se llama a `Results.gotoPreviousSlide()`, éste no tendrá ningún efecto, puesto que la diapositiva `Results` no contiene la diapositiva actual (es decir, el valor de `Results.currentSlide` es `null`).

Véase también

`Slide.currentSlide`, `Slide.gotoNextSlide()`, `Slide.previousSlide`

Slide.gotoSlide()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.gotoSlide(*newSlide*)

Parámetros

newSlide Diapositiva hasta la que se desplaza.

Valor devuelto

Valor booleano que indica si se ha desplazado correctamente (`true`) o no (`false`).

Descripción

Método; se desplaza hasta la diapositiva especificada por *newSlide*. Para desplazarse correctamente, deben darse las condiciones siguientes:

- La diapositiva actual debe ser secundaria de *mySlide*.
- La diapositiva especificada por *newSlide* y la diapositiva actual deben compartir una diapositiva ascendiente común, es decir, la diapositiva actual y *newSlide* deben residir en el mismo subárbol de la diapositiva.

Si no se cumple una de estas condiciones, el desplazamiento no se llevará a cabo correctamente y el método devolverá `false`; de lo contrario, el método se desplazará hasta la diapositiva especificada y devolverá `true`.

Por ejemplo, observe la siguiente jerarquía de diapositivas:

```
Presentation
  Slide1
    Slide1_1
    Slide1_2
 Slide2
    Slide2_1
    Slide2_2
```

Si la diapositiva actual es `Slide1_2`, la llamada al método `gotoSlide()` siguiente no se podrá realizar, puesto que la diapositiva actual no es descendiente de `Slide2`:

```
Slide2.gotoSlide(Slide2_1);
```

Observe también la siguiente jerarquía de pantallas, en la que un objeto de formulario es la pantalla principal de dos árboles de diapositiva distintos:

```
Form_1
  Slide1
    Slide1_1
    Slide1_2
  Slide2
    Slide2_1
    Slide2_2
```

Si la diapositiva actual es `Slide1_2`, la llamada al método siguiente tampoco se podrá realizar, puesto que `Slide1` y `Slide2` se encuentran en subárboles de diapositivas diferentes:

```
Slide1_2.gotoSlide(Slide2_2);
```

`Ejemplo`

El código siguiente, asociado con el componente Button, utiliza la propiedad `Slide.currentSlide` y el método `gotoSlide()` para mostrar la diapositiva siguiente en la presentación.

```
on(click) {
  _parent.gotoSlide(_parent.currentSlide.nextSlide);
}
```

Esto es equivalente al código siguiente, en el que se utiliza la función `Slide.gotoNextSlide()`:

```
on(click) {
  _parent.currentSlide.gotoNextSlide();
}
```

Véase también

`Slide.currentSlide`, `Slide.gotoNextSlide()`

Slide.hideChild

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(hideChild) {
  // El código se escribe aquí.
}
```

Descripción

Evento; se difunde cada vez que un elemento secundario de una diapositiva pasa de ser visible a invisible. Este evento sólo lo difunden las diapositivas, no los formularios. El uso principal del evento `hideChild` es aplicar transiciones de salida a todos los elementos secundarios de una diapositiva.

Ejemplo

Al asociarlo con la diapositiva raíz (por ejemplo, la diapositiva de presentación), este código mostrará el nombre de cada una de las diapositivas secundarias dependientes de la diapositiva raíz a medida que se vayan ocultando.

```
on(hideChild) {
  var child = eventObj.target._name;
  trace(child + " has just been hidden");
}
```

Véase también

`Slide.revealChild`

Slide.indexInParentSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.indexInParent`

Descripción

Propiedad (sólo lectura); devuelve el índice basado en cero de *`mySlide`* en la lista de diapositivas secundarias de su diapositiva principal.

Ejemplo

El código siguiente utiliza las propiedades `indexInParentSlide` y `Slide.numChildSlides` para mostrar el índice de la diapositiva actual que aparece en pantalla y el número total de diapositivas que contiene la diapositiva principal. Para utilizar este código, asócielo a una diapositiva principal que contenga una o varias diapositivas secundarias.

```
on (revealChild) {
  trace("Displaying "+(currentSlide.indexInParentSlide+1)+" of
  "+currentSlide._parent.numChildSlides);
}
```

Tenga en cuenta que, puesto que esta propiedad es un índice basado en cero, su valor aumenta en uno (`currentSlide.indexInParent+1`) para mostrar valores más significativos.

Véase también

`Slide.numChildSlides`, `Slide.revealChild`

Slide.lastSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.lastSlide`

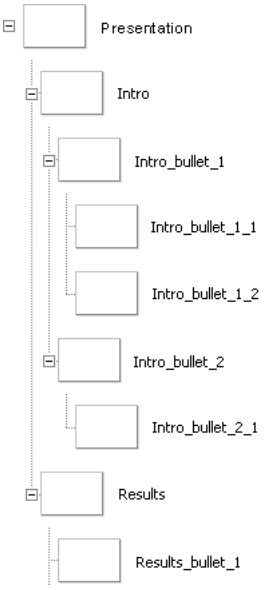
Descripción

Propiedad (sólo lectura); devuelve la última diapositiva secundaria de *`mySlide`* que no tiene diapositivas secundarias.

Ejemplo

Todas las sentencias siguientes relativas a la jerarquía de diapositivas son verdaderas:

```
Presentation.lastSlide._name == Results_bullet_1;
Intro.lastSlide._name == Intro_bullet_1_2;
Intro_bullet_1.lastSlide._name == Intro_bullet_1_2;
Results.lastSlide._name = Results_bullet_1;
```



Slide.nextSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.nextSlide

Descripción

Propiedad (sólo lectura); devuelve la diapositiva a la que se llegaría si se llama a *mySlide*.gotoNextSlide(), pero sin que se desplace realmente a dicha diapositiva. Por ejemplo, puede utilizar esta propiedad para mostrar el nombre de la diapositiva siguiente de una presentación y permitir a los usuarios que seleccione si desean desplazarse hasta dicha diapositiva.

Ejemplo

En este ejemplo, la etiqueta del componente Button denominado `nextButton` muestra el nombre de la diapositiva siguiente de la presentación. Si no hay más diapositivas, es decir si `mySlide.nextSlide` es `null`, la etiqueta del botón se actualizará para indicar al usuario que se encuentra al final de la presentación de diapositivas.

```
if (mySlide.nextSlide != null) {
    nextButton.label = "Next slide: " + mySlide.nextSlide._name + " > ";
} else {
  nextButton.label = "End of this slide presentation.";
}
```

Véase también

Slide.gotoNextSlide(), Slide.previousSlide

Slide.numChildSlides

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.numChildSlides

Descripción

Propiedad (sólo lectura); devuelve el número de diapositivas secundarias que contiene *mySlide*. Una diapositiva puede contener formularios u otras diapositivas; si *mySlide* contiene diapositivas y formularios, esta propiedad sólo devolverá el número de diapositivas y no contará los formularios.

Ejemplo

En este ejemplo se utiliza `Slide.numChildSlides` y el método `Slide.getChildSlide()` para repetir todas las diapositivas secundarias de la diapositiva de presentación raíz. A continuación, se muestran los nombres en el panel Salida.

```
var numSlides = _root.Presentation.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
  var childSlide = _root.Presentation.getChildSlide(slideIndex);
  trace(childSlide._name);
}
```

Véase también

`Slide.getChildSlide()`

Slide.overlayChildren

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide`.overlayChildren`

Descripción

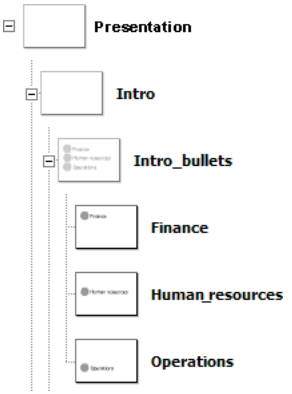
Propiedad; determina si las diapositivas secundarias de *mySlide* permanecen visibles al desplazarse de una diapositiva secundaria a la siguiente. Si el valor de esta propiedad es `true`, la diapositiva anterior permanece visible cuando se pase a la diapositiva siguiente del mismo nivel; si el valor es `false`, la diapositiva anterior quedará oculta cuando se pase a la diapositiva siguiente del mismo nivel.

Establecer esta propiedad en `true` resulta útil, por ejemplo, cuando una diapositiva determinada contiene varias diapositivas secundarias de "punto de viñeta" que se muestran por separado (posiblemente mediante transiciones); sin embargo, todas deben estar visibles cuando aparezcan los puntos de viñeta nuevos.

NOTA	Esta propiedad se aplica sólo a los descendientes inmediatos de *mySlide*, no a todas las diapositivas secundarias (anidadas).

Ejemplo

Por ejemplo, la diapositiva Intro_bullets de la ilustración siguiente contiene tres diapositivas secundarias (Finance, Human_resources y Operations), cada una de las cuales muestra un punto de viñeta independiente. Al establecer `Intro_bullets.overlayChildren` en `true`, cada diapositiva de viñeta permanecerá en el escenario a medida que vayan apareciendo los demás puntos de viñetas.



Slide.parentIsSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.parentIsSlide`

Descripción

Propiedad (sólo lectura); valor booleano que indica si el objeto principal de *`mySlide`* también es una diapositiva. Si el objeto principal de *`mySlide`* es una diapositiva, o pertenece a una subclase de Slide, esta propiedad devuelve `true`; de lo contrario, devuelve `false`.

Si *`mySlide`* es la diapositiva raíz de una presentación, esta propiedad devolverá `false` puesto que la diapositiva principal de la diapositiva de presentación es la principal (_level0) y no una diapositiva. Esta propiedad también devolverá `false` si el elemento principal de *`mySlide`* es un formulario.

Ejemplo

El código siguiente determina si el objeto principal de la diapositiva `mySlide` es, en sí mismo, una diapositiva. Si el valor de `mySlide.parentIsSlide` es `true`, el número de diapositivas del mismo nivel de `mySlide` aparecerá en el panel Salida. Si el objeto principal no es una diapositiva, Flash supone que `mySlide` es la diapositiva raíz (maestra) de la presentación y que, por lo tanto, no tiene dispositivas del mismo nivel.

```
if (mySlide.parentIsSlide) {
  trace("I have " + mySlide._parent.numChildSlides+" sibling slides");
} else {
  trace("I am the root slide and have no siblings");
}
```

Véase también

`Slide.numChildSlides`

Slide.parentSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.parentSlide`

Descripción

Propiedad (sólo lectura); referencia a la diapositiva que contiene la diapositiva actual.

Slide.playHidden

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.playHidden`

Descripción

Propiedad; valor booleano que especifica si *mySlide* debe continuar reproduciéndose una vez que se haya ocultado. Si está propiedad está establecida en `true`, *mySlide* continuará reproduciéndose cuando esté oculta. Si está establecida en `false`, *mySlide* dejará de reproducirse cuando se oculte; cuando aparezca de nuevo, volverá a reproducirse en el fotograma 1 de *mySlide*.

Slide.previousSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

mySlide.previousSlide

Descripción

Propiedad (sólo lectura); devuelve la diapositiva a la que se llegaría si se llama a *mySlide*.gotoPreviousSlide(), pero sin que se desplace realmente a dicha diapositiva. Por ejemplo, puede utilizar esta propiedad para mostrar el nombre de la diapositiva anterior de la presentación y permitir a los usuarios seleccionar si desean navegar hasta dicha diapositiva.

Ejemplo

En este ejemplo, la etiqueta del componente Button denominado `previousButton` muestra el nombre de la diapositiva anterior de la presentación. Si no hay ninguna diapositiva anterior, es decir si el valor de `mySlide.previousSlide` es `null`, la etiqueta del botón se actualizará para indicar al usuario que se encuentra al principio de la presentación de diapositivas.

```
if (mySlide.previousSlide != null) {
    previousButton.label = "Previous slide: " + mySlide.previous._name + "
  > ";
} else {
    previousButton.label = "You're at the beginning of this slide
  presentation.";
}
```

Véase también

Slide.gotoPreviousSlide(), Slide.nextSlide

Slide.revealChild

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
on(revealChild) {
  // El código se escribe aquí.
}
```

Descripción

Evento; se difunde cada vez que una diapositiva secundaria de un objeto de diapositiva pasa de ser invisible a visible. Este evento se utiliza principalmente para asociar transiciones "de entrada" con todas las diapositivas secundarias de una diapositiva determinada.

Ejemplo

Si se asocia con la diapositiva raíz, por ejemplo, la diapositiva de presentación, este código mostrará el nombre de todas las diapositivas secundarias a medida que vayan apareciendo.

```
on(revealChild) {
  var child = eventObj.target._name;
  trace(child + " has just appeared");
}
```

Véase también

`Slide.hideChild`

Slide.rootSlide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`mySlide``.rootSlide`

Descripción

Propiedad (sólo lectura); devuelve la diapositiva raíz del árbol o subárbol de diapositivas que contiene *mySlide*.

Ejemplo

Suponga que tiene un clip de película en una diapositiva y que al hacer clic en el mismo se accede a la primera diapositiva de la presentación. Para que esto suceda, debe asociar el código siguiente con el clip de película:

```
on(press) {
  _parent.rootSlide.gotoFirstSlide();
}
```

En este caso, `_parent` se refiere a la diapositiva que contiene el objeto de clip de película.

CAPÍTULO 43

Clase StyleManager

43

Nombre de clase de ActionScript mx.styles.StyleManager

La clase StyleManager realiza el seguimiento de los colores y estilos que se heredan. Sólo necesitará utilizar esta clase si ha creado componentes y desea añadir un color o un estilo heredado.

Para determinar qué estilos se heredan, consulte el sitio Web de W3C en www.w3.org/Style/CSS/.

Resumen de métodos de la clase StyleManager

En la tabla siguiente se enumeran los métodos de la clase StyleManager.

Método	Descripción
`StyleManager.registerColorName()`	Registra un nombre de color nuevo con StyleManager.
`StyleManager.registerColorStyle()`	Añade un estilo de color nuevo a StyleManager.
`StyleManager.registerInheritingStyle()`	Registra un estilo heredado nuevo con StyleManager.

StyleManager.registerColorName()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerColorName(colorName, value)
```

Parámetros

colorName Cadena que indica el nombre del color (por ejemplo, "gray", "darkGrey", y así sucesivamente).

value Número hexadecimal que indica el color (por ejemplo, 0x808080, 0x404040, y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; asocia un nombre de color con un valor hexadecimal y lo registra con Style Manager.

Ejemplo

En el ejemplo siguiente se registra "gray" como el nombre de color representado por el valor hexadecimal 0x808080:

```
StyleManager.registerColorName("gray", 0x808080);
```

StyleManager.registerColorStyle()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerColorStyle(colorStyle)
```

Parámetros

colorStyle Cadena que indica el nombre del color (por ejemplo, "highlightColor", "shadowColor", "disabledColor", y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; añade un estilo de color nuevo a Style Manager.

Ejemplo

En el ejemplo siguiente se registra `"highlightColor"` como un estilo de color:

```
StyleManager.registerColorStyle("highlightColor");
```

StyleManager.registerInheritingStyle()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerInheritingStyle(propertyName)
```

Parámetros

`propertyName` Cadena que indica el nombre de la propiedad de estilo (por ejemplo, `"newProp1"`, `"newProp2"`, y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; marca la propiedad de estilo como heredada. Utilice este método para registrar las propiedades de estilo que no aparecen en la lista de la especificación CSS. No utilice este método para cambiar propiedades de estilo de no heredadas a heredadas.

Si un valor de estilo no se hereda, sólo puede establecerse su estilo en una instancia y no en una hoja de estilos global o personalizada. Un estilo que no hereda su valor se define en la hoja de estilos de la clase, de forma que no es posible establecerlo en una hoja de estilos global o personalizada.

Ejemplo

En el ejemplo siguiente se registra `newProp1` como un estilo heredado:

```
StyleManager.registerInheritingStyle("newProp1");
```

CAPÍTULO 44

Clase SystemManager

44

Nombre de clase de ActionScript mx.managers.SystemManager

La clase SystemManager funciona automáticamente con la clase FocusManager con el fin de controlar cuál es la ventana de nivel superior que se activa en una aplicación que contiene componentes de la versión 2. También proporciona una propiedad `screen` que permite el acceso de los componentes y clips de película a las coordenadas del escenario.

Resumen de propiedades de la clase SystemManager

En la siguiente tabla se enumeran las propiedades de la clase SystemManager.

Propiedad	Descripción
`SystemManager.screen`	Sólo lectura; objeto que contiene el tamaño y la posición del escenario.

SystemManager.screen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
SystemManager.screen
```

Descripción

Propiedad; objeto con las propiedades `x`, `y`, `width` y `height`, que indica el tamaño y la posición del escenario.

Ejemplo

Si se establece un valor de `Stage.align` distinto de `"LT"`, es difícil saber qué coordenadas podrán verse realmente.

Suponga que desea colocar un clip de película de marca de agua (similar a las marcas de agua que utilizan muchos canales de televisión) en la esquina inferior derecha del escenario. El siguiente código funcionaría en todas las alineaciones del escenario correspondientes a la instancia de clip de película `watermark`:

```
import mx.managers.SystemManager;

var p1:Number = SystemManager.screen.width + SystemManager.screen.x -
  watermark._width;
var p2:Number = SystemManager.screen.height + SystemManager.screen.y -
  watermark._height;

watermark._x = p1;
watermark._y = p2;
```

CAPÍTULO 45

Componente TextArea

45

El componente TextArea ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para personalizar el componente TextArea; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo disabledColor. El componente TextArea también se puede formatear con HTML o como un campo de contraseña que disfraza el texto. Consulte "Aplicación de una hoja de estilos a un componente TextArea" en *Aprendizaje de ActionScript 2.0 en Flash*.

Es posible activar o desactivar el componente TextArea en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto TextField de ActionScript. Cuando la instancia de TextArea esté seleccionada, podrá controlarla con las teclas siguientes:

Tecla	Descripción
Teclas de flecha	Desplaza el punto de inserción una línea hacia arriba, abajo, izquierda o derecha.
Av Pág	Avanza una página.
Re Pág	Retrocede una página.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes* o "Clase FocusManager" en la página 745.

La previsualización dinámica de cada instancia de TextArea refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes. Si se necesita una barra de desplazamiento, ésta aparece en la previsualización dinámica, aunque no funciona. No es posible seleccionar texto en la previsualización dinámica, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente TextArea a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

Utilización del componente TextArea

Puede utilizar un componente TextArea siempre que necesite un campo de texto de varias líneas. Si necesita un campo de texto con una sola línea, utilice el Componente TextInput. Por ejemplo, suponga que precisa un componente TextArea como campo de comentarios de un formulario. En tal caso, puede definir un detector que compruebe si el campo está vacío cuando un usuario emplea el tabulador fuera del campo. El detector podría mostrar un mensaje de error indicando que debe introducirse un comentario en el campo.

Parámetros de TextArea

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente TextArea en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

editable indica si el componente TextArea es editable (`true`) o no (`false`). El valor predeterminado es `true`.

html indica si el texto va a formatearse con HTML (`true`) o no (`false`). SI HTML está definido en `true`, puede dar formato al texto con la etiqueta de fuente. El valor predeterminado es `false`.

text indica el contenido del componente TextArea. No es posible introducir retornos de carro en el inspector de propiedades ni en el inspector de componentes. El valor predeterminado es "" (una cadena vacía).

wordWrap indica si el texto se ajusta (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Si se crea un componente TextArea con el método createClassObject(), el valor predeterminado para wordWrap es `false`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente TextArea en el inspector de componentes (Ventana > Inspector de componentes):

maxChars es el número máximo de caracteres que puede contener el área de texto. El valor predeterminado es `null` (ilimitado).

restrict indica el conjunto de caracteres que el usuario puede introducir en el área de texto. El valor predeterminado es `undefined`. Véase "TextArea.restrict" en la página 1238.

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

password es un valor booleano que indica si la entrada es una contraseña u otro texto que debiera ocultarse mientras se escribe. Flash oculta los caracteres de entrada con asteriscos. El valor predeterminado es `false`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Es posible escribir código ActionScript para controlar éstas y otras opciones adicionales para el componente TextArea mediante sus propiedades, métodos y eventos. Para más información, consulte "Clase TextArea" en la página 1221.

Creación de aplicaciones con el componente TextArea

El procedimiento siguiente explica cómo añadir un componente TextArea a una aplicación durante la edición. En el ejemplo se configura un controlador de eventos `focusOut` en la instancia de TextArea que verifica que el usuario ha escrito algo en el área de texto antes de seleccionar una parte diferente de la interfaz.

Para crear una aplicación con el componente TextArea:

1. Arrastre un componente TextArea desde el panel Componentes al escenario y asígnele el nombre de instancia **my_ta**.
2. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

var taListener:Object = new Object();
taListener.focusOut = function(evt_obj:Object) {
 if (my_ta.length < 1) {
  trace("Please enter a comment");
 }
};
my_ta.addEventListener("focusOut", taListener);
```

Este código establece un controlador de eventos `focusOut` en la instancia de componente TextArea para comprobar que el usuario ha introducido algo en el área de texto.

Es posible obtener el valor del texto que se ha introducido en la instancia de TextArea de la siguiente manera:

```
var ta_text:String = my_ta.text;
```

Personalización del componente TextArea

El componente TextArea puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de la Clase TextArea.

Cuando se cambia el tamaño de un componente TextArea, el borde se ajusta al nuevo recuadro de delimitación. Si es necesario, las barras de desplazamiento se sitúan en los bordes inferior y derecho. A continuación, se ajusta el tamaño del área de texto al interior del área restante; el componente TextArea carece de elementos con tamaño fijo. Si el componente TextArea es demasiado pequeño para mostrar el texto, éste se recorta.

Utilización de estilos con el componente TextArea

El componente TextArea tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase sustituyen los estilos globales; por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración de estilo personalizada distinta en la instancia.

Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

Un componente TextArea admite los siguientes estilos:

Estilo	Tema	Descripción
`backgroundColor`	Ambos	Color del fondo. El color predeterminado es blanco.
`borderStyle`	Ambos	El componente TextArea utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es `"inset"`.
`marginLeft`	Ambos	Número que indica el margen izquierdo del texto. El valor predeterminado es 0.
`marginRight`	Ambos	Número que indica el margen derecho del texto. El valor predeterminado es 0.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"`, `"center"` o `"justify"`. (El parámetro `"justify"` sólo se admite en Flash Player 8). El valor predeterminado es `"left"`.

Estilo	Tema	Descripción
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.

Los componentes TextArea y TextInput utilizan exactamente los mismos estilos y suelen utilizarse del mismo modo. Por tanto, comparten de forma predeterminada la misma declaración de estilo de clase.

Por ejemplo, en el siguiente código se establece un estilo en la declaración de TextInput, pero afecta tanto al componente TextInput como al componente TextArea.

```
_global.styles.TextInput.setStyle("disabledColor", 0xBBBBFF);
```

Para separar los componentes y proporcionar estilos de clase de forma independiente, cree una nueva declaración de estilo.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.TextArea = new CSSStyleDeclaration();
_global.styles.TextArea.setStyle("disabledColor", 0xFFBBBB);
```

En este ejemplo no se comprueba si existía `_global.styles.TextArea` antes de sobrescribirlo; se supone que el usuario sabe que existe y desea sobrescribirlo.

Para establecer un fondo transparente en los componentes TextArea, defina de forma global el estilo `backgroundColor` con el valor `undefined`. A continuación, deberá definir el estilo `backgroundColor` con un color determinado en los casos en los que no desee que el fondo del componente TextArea sea transparente.

```
// Aplicar fondos transparentes a todos los componentes TextArea.
_global.styles.TextArea.backgroundColor = undefined;

// Aplicar un fondo blanco a esta instancia específica del componente.
myTextArea2.setStyle( "backgroundColor", "white" );
```

El componente TextArea admite un conjunto de estilos de componente para todo el texto del campo. Sin embargo, también puede mostrar código HTML compatible con la representación HTML de Flash Player. Para mostrar texto HTML, defina `TextArea.html` en `true`.

Si define el componente TextArea para que muestre texto HTML, el estilo de texto se define con la clase TextField.StyleSheet (para más información detallada sobre esta clase, consulte la *Referencia del lenguaje ActionScript 2.0*). Por ejemplo:

1. Arrastre un componente TextArea al escenario y asígnele el nombre de instancia `my_ta`.

2. Introduzca este código en el panel Acciones del fotograma 1 de la línea de tiempo:

```
var my_styles = new TextField.StyleSheet();
my_styles.setStyle("p", {fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'12px', color:'#CC6699'});
my_ta.styleSheet = my_styles;
my_ta.html = true;
my_ta.text = "<p>This is some text</p>";
```

Utilización de aspectos con el componente TextArea

El componente TextArea utiliza una instancia de RectBorder para su borde y barras de desplazamiento para desplazarse por las imágenes. Para más información sobre la aplicación de aspectos en estos elementos visuales, consulte "Clase RectBorder" en la página 1103 y "Utilización de aspectos con el componente UIScrollBar" en la página 1435.

Clase TextArea

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > TextArea

Nombre de clase de ActionScript mx.controls.TextArea

Las propiedades de la clase TextArea permiten definir el contenido y el formato del texto, así como las posiciones vertical y horizontal en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo de contraseña. o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase TextArea se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente TextArea sustituye el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas.

El componente TextArea admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.TextArea.version);
```

NOTA El código `trace(myTextAreaInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase TextArea

No hay métodos exclusivos de la clase TextArea.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase TextArea de la clase UIObject. Al llamar a estos métodos desde el objeto TextArea, debe utilizarse la forma *TextAreaInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase TextArea de la clase UIComponent. Al llamar a estos métodos desde el objeto TextArea, debe utilizarse la forma *TextAreaInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase TextArea

En la tabla siguiente se enumeran las propiedades de la clase TextArea.

Propiedad	Descripción
`TextArea.editable`	Valor booleano que indica si el campo es editable (`true`) o no (`false`).
`TextArea.hPosition`	Define la posición horizontal del texto en el campo.
`TextArea.hScrollPolicy`	Indica si la barra de desplazamiento horizontal está activada siempre (`"on"`), nunca (`"off"`) o se activa cuando es necesario (`"auto"`).
`TextArea.html`	Valor booleano que indica si el contenido del área de texto admite el formato HTML.
`TextArea.length`	Sólo lectura; número de caracteres del área de texto.
`TextArea.maxChars`	Número máximo de caracteres que puede contener el área de texto.
`TextArea.maxHPosition`	Sólo lectura; valor máximo de `TextArea.hPosition`.
`TextArea.maxVPosition`	Sólo lectura; valor máximo de `TextArea.vPosition`.
`TextArea.password`	Valor booleano que indica si el campo es un campo de contraseña (`true`) o no (`false`).
`TextArea.restrict`	Conjunto de caracteres que puede introducir un usuario en el área de texto.
`TextArea.styleSheet`	Asocia una hoja de estilos con el componente TextArea especificado.
`TextArea.text`	Contenido de texto de un componente TextArea.
`TextArea.vPosition`	Número que indica la posición de desplazamiento vertical.
`TextArea.vScrollPolicy`	Indica si la barra de desplazamiento vertical está activada siempre (`"on"`), nunca (`"off"`) o se activa cuando es necesario (`"auto"`).
`TextArea.wordWrap`	Valor booleano que indica si el texto se ajusta (`true`) o no (`false`).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase TextArea de la clase UIObject. Al acceder a estas propiedades desde el objeto TextArea, debe utilizarse la forma *TextAreaInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
UIObject.height	Sólo lectura; altura del objeto, expresada en píxeles.
UIObject.left	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.right	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
UIObject.scaleX	Sólo lectura; número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Sólo lectura; anchura del objeto, expresada en píxeles.
UIObject.x	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.y	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase TextArea de la clase UIComponent. Al acceder a estas propiedades desde el objeto TextArea, debe utilizarse la forma *TextAreaInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase TextArea

En la tabla siguiente se muestra el evento de la clase TextArea.

Evento	Descripción
TextArea.change	Notifica a los detectores que el texto ha cambiado.
TextArea.scroll	Notifica a los detectores que el texto se ha desplazado.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase TextArea de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase TextArea de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

TextArea.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
textAreaInstance.addEventListener("change", listenerObject);
```

Sintaxis 2:

```
on (change) {
  // ...
}
```

Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al área de texto del componente; para ello, deberá utilizar `TextArea.restrict`.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textAreaInstance*) distribuye un evento (en este caso, change) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de TextArea. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextArea`, envía "_level0.myTextArea" al panel Salida.

```
on (change) {
  trace(this);
}
```

Ejemplo

En este ejemplo se utiliza el modelo de evento distribuidor/detector para realizar un seguimiento del número total de veces que ha cambiado el área de texto en un componente TextArea denominado `my_ta`:

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

// Crear una variable Number para realizar un seguimiento del número de
  cambios de TextArea.
var changeCount_num:Number = 0;
```

```
// Definir un objeto detector.
var taListener:Object = new Object();
// Definir una función que se ejecuta cuando el detector recibe
// la notificación de un cambio en el componente TextArea.
taListener.change = function(evt_obj:Object) {
 changeCount_num++;
 trace("Text has changed " + changeCount_num + " times now!");
 trace("It now contains: " + evt_obj.target.text);
 trace("");
};
// Registrar el objeto detector con la instancia del componente TextArea.
my_ta.addEventListener("change", taListener);
```

Véase también

`EventDispatcher.addEventListener()`

TextArea.editable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance``.editable`

Descripción

Propiedad; valor booleano que indica si el componente es editable (`true`) o no (`false`). El valor predeterminado es `true`.

Ejemplo

En el siguiente ejemplo se establece la propiedad `editable` en `false` para evitar que el usuario pueda editar el texto que se carga en la instancia de TextArea denominada `my_ta`.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;
```

```
my_ta.setSize(320, 240);
my_ta.editable = false;

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.hPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.hPosition

Descripción

Propiedad; define en píxeles la posición horizontal del texto en el campo. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se utiliza un detector para mostrar la posición horizontal actual en el panel Salida mientras el usuario desplaza hacia atrás y hacia adelante el texto que se ha cargado en la instancia de TextArea denominada `my_ta`.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;
```

```
my_ta.setSize(320, 240);
my_ta.wordWrap = false;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("hPosition = " + my_ta.hPosition);
}
my_ta.addEventListener("scroll", taListener);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
  my_ta.hPosition = 200;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.hScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.hScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre (`"on"`), nunca (`"off"`) o aparece automáticamente en función del tamaño del campo (`"auto"`). El valor predeterminado es `"auto"`.

Ejemplo

En el siguiente ejemplo se desactiva la propiedad `hScrollPolicy`, lo que provoca que la instancia de TextArea no tenga una barra de desplazamiento.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;
my_ta.hScrollPolicy = "off";

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.html

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance``.html`

Descripción

Propiedad; valor booleano que indica si el contenido del área de texto tiene el formato HTML (`true`) o no (`false`). Si el valor de la propiedad `html` es `true`, el contenido del área de texto es HTML. Si el valor de `html` es `false`, el área de texto no es HTML. El valor predeterminado es `false`.

Ejemplo

En el ejemplo siguiente se convierte el componente TextArea denominado `my_ta` en un área de texto HTML y luego se formatea el texto con etiquetas HTML.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.html = true;
my_ta.text = "The <b>Royal</b> Nonesuch";
```

TextArea.length

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance``.length`

Descripción

Propiedad (sólo lectura); indica el número de caracteres de un área de texto. Esta propiedad devuelve el mismo valor que la propiedad `text.length` de ActionScript, pero es más rápida. El carácter de tabulador (`"\t"`) cuenta como un carácter. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se accede a la propiedad length para mostrar el número de caracteres que escribe el usuario en el componente TextArea denominado `my_ta`.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

// Definir un objeto detector.
var taListener:Object = new Object();
taListener.change = function(evt_obj:Object) {
 trace("my_ta.length is now: " + my_ta.length + " characters");
};
my_ta.addEventListener("change", taListener);
```

TextArea.maxChars

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance``.maxChars`

Descripción

Propiedad; indica el número máximo de caracteres que puede contener el área de texto. Un script puede insertar más texto del que permite la propiedad `maxChars`; la propiedad sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es `null`.

Ejemplo

En el siguiente ejemplo se establece la propiedad `maxchars` para limitar a 24 el número de caracteres que puede introducir un usuario.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.maxChars = 24;

// Definir un objeto detector.
var taListener:Object = new Object();
taListener.change = function(evt_obj:Object) {
 trace("my_ta.length is now: " + my_ta.length + " characters");
};
my_ta.addEventListener("change", taListener);
```

TextArea.maxHPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.maxHPosition

Descripción

Propiedad de sólo lectura; valor máximo de `TextArea.hPosition`. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se accede a la propiedad `maxHPosition` para definir la posición inicial del componente TextArea denominado `my_ta` en la posición derecha más alejada.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("hPosition = " + my_ta.hPosition);
}
my_ta.addEventListener("scroll", taListener);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
  my_ta.hPosition = my_ta.maxHPosition;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Véase también

`TextArea.vPosition`

TextArea.maxVPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.maxVPosition

Descripción

Propiedad de sólo lectura; indica el valor máximo de TextArea.vPosition. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se accede a la propiedad maxVPosition para definir la posición inicial vertical del componente TextArea denominado my_ta en la parte inferior. También traza la posición vertical actual mientras el usuario se desplaza hacia arriba y hacia abajo.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre my_ta; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
 - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = true;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("vPosition = " + my_ta.vPosition);
}
my_ta.addEventListener("scroll", taListener);

// Cargar texto para mostrar y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
     my_ta.vPosition = my_ta.maxVPosition;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Véase también

TextArea.hPosition

TextArea.password

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.password

Descripción

Propiedad; valor booleano que indica si el área de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de `password` es `true`, el área de texto es un área de contraseña y oculta los caracteres de entrada con asteriscos. Si el valor de `password` es `false`, el área de texto no es de contraseña. El valor predeterminado es `false`.

Ejemplo

En el siguiente ejemplo se considera el texto del componente TextArea denominado `my_ta` un campo de contraseña si se activa la casilla de verificación denominada`my_ch`. De lo contrario, se considera texto corriente.

En primer lugar, añada una instancia del componente TextArea al escenario, asígnele el nombre `my_ta`, añada una casilla de verificación y denomínela `my_ch`, a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
  - Instancia de CheckBox en el escenario (nombre de instancia: my_ch)
*/
var my_ta:mx.controls.TextArea;
var my_ch:mx.controls.CheckBox;

my_ta.wordWrap = false;
my_ta.password = true;
my_ch.selected = my_ta.password;

var chListener:Object = new Object();
chListener.click = function(evt_obj:Object) {
 my_ta.password = my_ch.selected;
}
my_ch.addEventListener("click", chListener);
```

TextArea.restrict

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.restrict

Descripción

Propiedad; indica el conjunto de caracteres que los usuarios pueden introducir en el área de texto. El valor predeterminado es `undefined`. Si el valor de esta propiedad es `null`, los usuarios puede introducir cualquier carácter. Si esta propiedad contiene una cadena vacía, no se puede introducir ningún carácter. Si esta propiedad contiene una cadena de caracteres, los usuarios sólo pueden introducir caracteres en la cadena; la cadena se explora de izquierda a derecha. Puede especificarse un rango utilizando un guión (-).

Si la cadena empieza por ^, no se acepta ninguno de los caracteres posteriores a ^. Si la cadena no empieza por ^, se aceptan los caracteres de la cadena. También es posible utilizar ^ para alternar entre los caracteres que se aceptan y los que no se aceptan.

Por ejemplo, en el código siguiente se permite A-Z excepto X y Q:

```
Ta.restrict = "A-Z^XQ";
```

Si se restringen las entradas a caracteres en mayúsculas, los caracteres alfabéticos que se introduzcan en minúsculas se convertirán en mayúsculas. Igualmente, si se restringen las entradas a caracteres en minúsculas, los caracteres que se introduzcan en mayúsculas se convertirán en minúsculas.

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el área de texto. Esta propiedad no se sincroniza con las casillas de verificación Incorporar contornos de fuentes del inspector de propiedades.

Ejemplo

En el siguiente ejemplo se establece, en primer lugar, la propiedad restrict para limitar el área de texto a letras en mayúsculas, números y espacios y, a continuación, se establece para permitir todos los caracteres excepto letras en minúsculas.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1. Utilice sólo una configuración para la propiedad `restrict` cada vez.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/
var my_ta:mx.controls.TextArea;

my_ta.wordWrap = true;

// Limitar control a letras en mayúsculas, números y espacios.
my_ta.restrict = "A-Z 0-9";

// Permitir todos los caracteres, excepto letras en minúsculas
// caracteres en mayúscula
my_ta.restrict = "^a-z";
```

TextArea.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
textAreaInstance.addEventListener("scroll", listenerObject);
```

Sintaxis 2:

```
on (scroll) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el botón del ratón (se suelta) sobre la barra de desplazamiento. La propiedad `UIScrollBar.scrollPosition` y la imagen en pantalla de la barra de desplazamiento se actualizan antes de que se difunda este evento.

El primer ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector en el que el script se sitúa en un fotograma de la línea de tiempo que contiene la instancia del componente. Una instancia de componente (*`textAreaInstance`*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se produce el evento. Cuando se produce el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama a `addEventListener()` (véase `EventDispatcher.addEventListener()`) en la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Además de las propiedades normales del objeto de evento (`type` y `target`), el objeto del evento `scroll` incluye una tercera propiedad denominada `direction`. La propiedad `direction` contiene una cadena que describe la orientación de la barra de desplazamiento. Los valores posibles de la propiedad `direction` son `vertical` (valor predeterminado) y `horizontal`.

Para más información sobre las propiedades `type` y `target` del objeto de evento, consulte "Objetos de evento" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente TextArea. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia del componente TextArea `myTextAreaComponent`, envía "_level0.myTextAreaComponent" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En este ejemplo se utiliza el modelo de evento distribuidor/detector para realizar un seguimiento del momento en que el usuario se desplaza por el componente TextArea con las barras de desplazamiento o el cuadro de desplazamiento de la instancia de TextArea.

En primer lugar, añada una instancia del componente TextArea al escenario y asígnele el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

my_ta.setSize(320, 240);
my_ta.move(10, 10);

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  my_ta.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

my_ta.addEventListener("scroll", doScroll);
function doScroll(evt_obj:Object):Void {
  trace("target:    " + evt_obj.target);
  trace("type:      " + evt_obj.type);
  trace("direction: " + evt_obj.direction);
  trace("position:  " + evt_obj.position);
  trace("");
}
```

Véase también

`EventDispatcher.addEventListener()`

TextArea.styleSheet

Disponibilidad

Flash Player 7.

Sintaxis

`textAreaInstance``.styleSheet = `*`TextFieldStyleSheetObject`*

Descripción

Propiedad; asocia una hoja de estilos con el componente TextArea especificado por *`TextAreaInstance`*. Para más información sobre la creación de hojas de estilos, consulte "Aplicación de formato al texto con hojas de estilos en cascada" en *Aprendizaje de ActionScript 2.0 en Flash*.

La hoja de estilos asociada con un componente TextArea puede cambiar en cualquier momento. Si se cambia la hoja de estilos en uso, el componente TextArea vuelve a dibujarse con la nueva hoja de estilos. Para eliminar la hoja de estilos, puede definirse en `null` o `undefined`. Si se elimina la hoja de estilos en uso, el componente TextArea vuelve a dibujarse sin ninguna hoja de estilos. El formato aplicado por una hoja de estilos no se conserva si se elimina la hoja de estilos.

Ejemplo

En el código siguiente se crea un nuevo objeto StyleSheet denominado `my_styles` con el constructor `new TextField.StyleSheet`. A continuación, se definen los estilos para las etiquetas `html` y `body`. Después, se aplica el estilo asignando `my_styles` a la propiedad `styleSheet` de la instancia de TextArea `my_ta`.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);

// Crear el nuevo objeto StyleSheet.
var my_styles:TextField.StyleSheet = new TextField.StyleSheet();
my_styles.setStyle("html", {fontFamily:"Arial,Helvetica,sans-serif",
  fontSize:"12px", color:"#0000FF"});
my_styles.setStyle("body", {color:"#00CCFF", textDecoration:"underline"});

// Establecer la propiedad TextAreaInstance.styleSheet en el nuevo
// objeto styleSheet denominado styles.
my_ta.styleSheet = my_styles;
my_ta.html = true;

// Cargar el texto que debe mostrarse y definir controlador onLoad.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
 if (src != undefined) {
  my_ta.text = src;
 } else {
  my_ta.text = "Error loading HTML document.";
 }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.text

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.text

Descripción

Propiedad; contenido de texto de un componente TextArea. El valor predeterminado es "" (una cadena vacía).

Ejemplo

En el siguiente ejemplo se coloca una cadena en la propiedad `text` de la instancia `my_ta` `TextArea` y, a continuación, se rastrea esa cadena en el panel Salida.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.text = "The Royal Nonesuch";
trace(my_ta.text); // traza "The Royal Nonesuch"
```

TextArea.vPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.vPosition

Descripción

Propiedad; define la posición de desplazamiento vertical del texto en un área de texto. Esta propiedad es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo, o para crear áreas de texto desplazable. Es posible obtener y definir esta propiedad. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se carga el texto en el componente TextArea denominado `my_ta` y se define la propiedad vPosition para que muestre el texto en la parte inferior.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
 if (src != undefined) {
  my_ta.text = src;
  my_ta.vPosition = my_ta.maxVPosition;
 } else {
  trace("Error loading text.");
 }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt")
```

TextArea.vScrollPolicy

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.vScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre (`"on"`), nunca (`"off"`) o aparece automáticamente en función del tamaño del campo (`"auto"`). El valor predeterminado es `"auto"`.

Ejemplo

En el siguiente ejemplo se desactiva la barra de desplazamiento vertical del componente TextArea denominado my_ta, de forma que no haya barra de desplazamiento para desplazarse por el texto que carga el ejemplo.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre `my_ta`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/
var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = true;
my_ta.vScrollPolicy = "off";

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.wordWrap

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance``.wordWrap`

Descripción

Propiedad; valor booleano que indica si el texto se ajusta (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Si se crea un componente TextArea con el método `createClassObject()`, el valor predeterminado para wordWrap es `false`.

Ejemplo

En el siguiente ejemplo se establece la propiedad `wordwrap` en `false` del componente TextArea denominado `my_ta`, lo que hace que haya una barra de desplazamiento horizontal para acceder al texto que está fuera de los límites laterales.

En primer lugar, debe añadir una instancia del componente TextArea al escenario y asignarle el nombre **my_ta**; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextArea en el escenario (nombre de instancia: my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;
//my_ta.vScrollPolicy = "off";

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

CAPÍTULO 46

Componente TextInput

46

TextInput es un componente de una sola línea de texto que ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para personalizar el componente TextInput; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo disabledColor. El componente TextInput se puede formatear también con HTML o como un campo de contraseña que disfraza el texto.

Es posible activar o desactivar el componente TextInput en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto TextField de ActionScript. Cuando la instancia de TextInput está seleccionada, puede controlarla con las teclas siguientes:

Tecla	Descripción
Teclas de flecha	Desplaza el punto de inserción un carácter hacia la izquierda o la derecha.
Mayús+Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

La previsualización dinámica de cada instancia de TextInput refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el inspector de componentes. No es posible seleccionar texto en la previsualización dinámica, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente TextInput a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

Utilización del componente TextInput

Puede utilizar un componente TextInput siempre que necesite un campo de texto de una sola línea. Si necesita un campo de texto con varias líneas, utilice el Componente TextArea. Por ejemplo, si desea utilizar un componente TextInput como campo de contraseña en un formulario, puede definir un detector que compruebe si el campo tiene caracteres suficientes cuando el usuario utiliza el tabulador para salir del campo. El detector podría mostrar un mensaje de error indicando que se debe introducir el número de caracteres correcto.

Parámetros de TextInput

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente TextInput en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

editable indica si el componente TextInput es editable (`true`) o no (`false`). El valor predeterminado es `true`.

password indica si el campo es un campo de contraseña (`true`) o no (`false`). El valor predeterminado es `false`.

text especifica el contenido del componente TextInput. No es posible introducir retornos de carro en el inspector de propiedades ni en el inspector de componentes. El valor predeterminado es "" (una cadena vacía).

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente TextInput en el inspector de componentes (Ventana > Inspector de componentes):

maxChars número máximo de caracteres que puede contener el campo de entrada de texto. El valor predeterminado es `null` (ilimitado).

restrict indica el conjunto de caracteres que el usuario puede introducir en el campo de entrada de texto. El valor predeterminado es `undefined`. Véase "TextInput.restrict" en la página 1267.

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Es posible escribir código ActionScript para controlar éstas y otras opciones adicionales para el componente TextInput mediante sus propiedades, métodos y eventos. Para más información, consulte "Clase TextInput" en la página 1252.

Creación de aplicaciones con el componente TextInput

El procedimiento siguiente explica cómo añadir un componente TextInput a una aplicación durante la edición. En este ejemplo, el componente es un campo de contraseña con un detector de eventos que determina si se ha introducido el número de caracteres correcto.

Para crear una aplicación con el componente TextInput:

1. Arrastre un componente TextInput desde el panel Componentes al escenario.
2. En el inspector de propiedades, siga este procedimiento:
   - Introduzca el nombre de instancia **my_ti**.
   - Deje el parámetro text vacío.
   - Defina el parámetro editable en `true`.
3. Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.handleEvent = function (evt_obj:Object){
 if (evt_obj.type == "enter"){
  if (my_ti.length < 8) {
   trace("You must enter at least 8 characters");
  } else {
   trace("Thanks");
  }
 }
}
// Añadir detector.
my_ti.addEventListener("enter", tiListener);
```

Este código establece un controlador de eventos `enter` en la instancia de TextInput denominada `my_ti`. Si el usuario escribe menos de ocho caracteres, el ejemplo muestra el mensaje: `You must enter at least 8 characters`. Si el usuario introduce ocho o más caracteres, el ejemplo muestra: `Thanks`.

4. Una vez introducido el texto en la instancia `my_ti`, puede obtener su valor como se muestra a continuación:

```
var my_text:String = my_ti.text;
```

Personalización del componente TextInput

El componente TextInput puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de la Clase TextInput.

Cuando se cambia el tamaño de un componente TextInput, el borde se ajusta al nuevo recuadro de delimitación. El componente TextInput no utiliza barras de desplazamiento, si bien el punto de inserción se desplaza automáticamente a medida que el usuario interactúa con el texto. Así, el tamaño del campo de texto se ajusta al interior del área restante; el componente TextInput carece de elementos con tamaño fijo. Si el componente TextInput es demasiado pequeño para mostrar el texto, éste se recorta.

Utilización de estilos con el componente TextInput

El componente TextInput tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase sustituyen los estilos globales; por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración de estilo personalizada distinta o definirla en la instancia.

Un componente TextInput admite los siguientes estilos:

Estilo	Tema	Descripción
`backgroundColor`	Ambos	Color del fondo. El color predeterminado es blanco.
`borderStyle`	Ambos	El componente TextInput utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase “Clase RectBorder” en la página 1103. El estilo de borde predeterminado es `"inset"`.

Estilo	Tema	Descripción
`marginLeft`	Ambos	Número que indica el margen izquierdo del texto. El valor predeterminado es 0.
`marginRight`	Ambos	Número que indica el margen derecho del texto. El valor predeterminado es 0.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es`false`.
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es `10`.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es`"none"`.

Los componentes TextArea y TextInput utilizan los mismos estilos y suelen utilizarse del mismo modo. Por tanto, comparten de forma predeterminada la misma declaración de estilo de clase. Por ejemplo, en el siguiente código se establece un estilo en la declaración de TextArea, pero afecta tanto al componente TextArea como al componente TextInput.

```
_global.styles.TextArea.setStyle("disabledColor", 0xBBBBFF);
```

Para separar los componentes y proporcionar estilos de clase de forma independiente, cree una nueva declaración de estilo.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.TextInput = new CSSStyleDeclaration();
_global.styles.TextInput.setStyle("disabledColor", 0xFFBBBB);
```

Observe que en este ejemplo no se comprueba si existía `_global.styles.TextInput` antes de sobrescribirlo; se supone que el usuario sabe que existe y desea sobrescribirlo.

Utilización de aspectos con el componente TextInput

El componente TextArea utiliza una instancia de RectBorder para definir el borde. Para más información sobre la aplicación de aspectos en estos elementos visuales, consulte "Clase RectBorder" en la página 1103.

Clase TextInput

Herencia MovieClip > Clase UIObject > Clase UIComponent > TextInput

Nombre de clase de ActionScript mx.controls.TextInput

Las propiedades de la clase TextInput permiten definir el contenido, el formato y la posición horizontal del texto en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo de contraseña. o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase TextInput se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el inspector de componentes.

El componente TextInput utiliza Focus Manager para sustituir el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte "Clase FocusManager" en la página 745.

El componente TextInput admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player. Para información sobre la compatibilidad con CSS, consulte la especificación W3C en www.w3.org/TR/REC-CSS2/.

Puede manipular la cadena de texto utilizando la cadena devuelta por el objeto de texto.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.TextInput.version);
```

NOTA	El código `trace(myTextInputInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase TextInput

No hay métodos exclusivos de la clase TextInput.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase TextInput de la clase UIObject. Al llamar a estos métodos desde el objeto TextInput, debe utilizarse la forma *`TextInputInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase TextInput de la clase UIComponent. Al llamar a estos métodos desde el objeto TextInput, debe utilizarse la forma *`TextInputInstance.methodName`*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Resumen de propiedades de la clase TextInput

En la tabla siguiente se enumeran las propiedades de la clase TextInput.

Propiedad	Descripción
`TextInput.editable`	Valor booleano que indica si el campo es editable (`true`) o no (`false`).
`TextInput.hPosition`	Posición de desplazamiento horizontal del texto en el campo.
`TextInput.length`	Sólo lectura; número de caracteres de un componente TextInput.
`TextInput.maxChars`	Número máximo de caracteres que puede introducir un usuario en el campo de texto.
`TextInput.maxHPosition`	Sólo lectura; valor máximo posible de `TextField.hPosition`.
`TextInput.password`	Valor booleano que indica si el campo de texto es un campo de contraseña que oculta los caracteres introducidos.
`TextInput.restrict`	Indica los caracteres que el usuario puede introducir en un campo de texto.
`TextInput.text`	Define el contenido de texto de componente TextInput.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase TextInput de la clase UIObject. Al acceder a estas propiedades desde el objeto TextInput, debe utilizarse la forma *TextInputInstance.propertyName*.

Propiedad	Descripción
UIObject.bottom	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
UIObject.height	Sólo lectura; altura del objeto, expresada en píxeles.
UIObject.left	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.right	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Sólo lectura; anchura del objeto, expresada en píxeles.
UIObject.x	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.y	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase TextInput de la clase UIComponent. Al acceder a estas propiedades desde el objeto TextInput, debe utilizarse la forma *TextInputInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase TextInput

En la tabla siguiente se enumeran los eventos de la clase TextInput.

Evento	Descripción
`TextInput.change`	Se difunde cuando se modifica el campo TextInput.
`TextInput.enter`	Se difunde cuando se presiona la tecla Intro.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase TextInput de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase TextInput de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

TextInput.change

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  //...
};
textInputInstance.addEventListener("change", listenerObject)
```

Sintaxis 2:

```
on (change){
  //...
}
```

Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al campo de texto del componente; para ello, deberá utilizar `TextInput.restrict`. Este evento se activa sólo por una entrada del usuario, no por un cambio provocado por ActionScript.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de TextInput. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextInput`, envía "_level0.myTextInput" al panel Salida:

```
on (change){
  trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector. Una instancia de componente (*textInputInstance*) distribuye un evento (en este caso, change) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método EventDispatcher.addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En este ejemplo se crea un detector para un evento change en la instancia de TextInput my_ti. Cuando se produce un evento change, el ejemplo muestra "Input has changed".

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia my_ti; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("Input has changed");
};
// Añadir detector.
my_ti.addEventListener("change", tiListener);
```

Véase también

EventDispatcher.addEventListener()

TextInput.editable

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.editable

Descripción

Propiedad; valor booleano que indica si el componente es editable (`true`) o no (`false`). El valor predeterminado es `true`.

Ejemplo

En este ejemplo se establece la propiedad `editable` en el valor `false` para la instancia my_ti de TextInput. Esto evita que el usuario introduzca texto en la instancia. Puede establecer la propiedad en `true` para hacer que la instancia de TextInput pueda editarse.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.editable = false;
```

TextInput.enter

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.enter = function(eventObject:Object) {
  //...
};
textInputInstance.addEventListener("enter", listenerObject);
```

Sintaxis 2:

```
on (enter) {
  //...
}
```

Descripción

Evento; notifica a los detectores que se ha presionado la tecla Intro.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de TextInput. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextInput`, envía "_level0.myTextInput" al panel Salida:

```
on (enter){
  trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector. Una instancia de componente (*textInputInstance*) distribuye un evento (en este caso, enter) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método EventDispatcher.addEventListener() de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En este ejemplo se crea un detector para un evento enter en una instancia de TextInput denominada my_ti. Cuando se produce el evento enter, si el usuario ha introducido menos de ocho caracteres, el ejemplo muestra: You must enter at least 8 characters. De lo contrario, muestra Thanks!

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia my_ti; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.handleEvent = function (evt_obj:Object){
 if (evt_obj.type == "enter"){
  if (my_ti.length < 8) {
   trace("You must enter at least 8 characters");
  } else {
   trace("Thanks");
  }
 }
}
// Añadir detector.
my_ti.addEventListener("enter", tiListener);
```

Véase también

EventDispatcher.addEventListener()

TextInput.hPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.hPosition

Descripción

Propiedad; especifica los píxeles que se han desplazado para que la entrada del usuario quepa en el cuadro de TextInput. El valor predeterminado es 0.

NOTA	El valor cambia para el mismo texto en equipos diferentes debido al monitor, al tamaño de pantalla y a las características de la fuente.

Ejemplo

En el siguiente ejemplo se crea un detector para un evento `change` en la instancia de TextInput denominada `my_ti`. El detector accede a la propiedad `hPosition` para mostrar la posición actual de cada uno de los caracteres que introduce el usuario.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("hPosition = " + my_ti.hPosition);
};
// Añadir detector.
my_ti.addEventListener("change", tiListener);
```

TextInput.length

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.length

Descripción

Propiedad de sólo lectura; valor que indica el número de caracteres de un componente TextInput. El carácter de tabulador ("\t") cuenta como un carácter. El valor predeterminado es 0.

Ejemplo

En el siguiente ejemplo se crea un detector para un evento change en la instancia de TextInput denominada my_ti. El detector accede a la propiedad length para mostrar la longitud del texto en my_ti mientras el usuario introduce el texto.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia my_ti; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("Length of text: " + my_ti.length);
};
// Añadir detector.
my_ti.addEventListener("change", tiListener);
```

TextInput.maxChars

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.maxChars

Descripción

Propiedad; indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto del que permite la propiedad `maxChars`; esta propiedad sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es `null`.

Ejemplo

En el siguiente ejemplo se limita a ocho el número de caracteres que puede introducir un usuario en la instancia de TextInput denominada `my_ti`. También se establece la propiedad `password`, que oculta los caracteres de entrada mostrando asteriscos en lugar de los caracteres introducidos.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.maxChars = 8;
my_ti.password = true;
```

TextInput.maxHPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.maxHPosition

Descripción

Propiedad de sólo lectura; el valor de `maxHPosition` es la posición en píxeles del carácter visible cuando el puntero se mueve totalmente a la derecha del texto. No es la posición en píxeles del último carácter. Más bien, es la posición en píxeles en el extremo derecho del último carácter en el campo TextInput. El valor predeterminado es 0.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

Ejemplo

El siguiente código crea un detector para un evento change en la instancia de TextInput denominada `my_ti`. Cuando se produce el evento change, el detector muestra los valores actuales de `hPosition` y `maxHPosition` para cada carácter que introduce el usuario:

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Crear un objeto detector.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("hPosition: " + my_ti.hPosition + " of " + my_ti.maxHPosition);
};
// Añadir detector.
my_ti.addEventListener("change", tiListener);
```

TextInput.password

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.password

Descripción

Propiedad; valor booleano que indica si el campo de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de esta propiedad es `true`, el campo de texto es un campo de contraseña y oculta los caracteres de entrada. Si su valor es `false`, el campo de texto no es de contraseña. El valor predeterminado es `false`.

Ejemplo

En el siguiente ejemplo se define la propiedad `password` para mostrar un asterisco en lugar del carácter que introduce el usuario en la instancia de TextInput denominada `my_ti`. También se define `maxChars` para limitar a ocho el número máximo de caracteres que puede introducir el usuario.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.maxChars = 8;
my_ti.password = true;
```

TextInput.restrict

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.restrict

Descripción

Propiedad; indica el conjunto de caracteres que el usuario puede introducir en el campo de texto. El valor predeterminado es `undefined`. Si el valor de esta propiedad es `null` o una cadena vacía (`""`), el usuario puede introducir cualquier carácter. Si esta propiedad contiene una cadena de caracteres, el usuario sólo puede introducir caracteres en la cadena; la cadena se explora de izquierda a derecha. Puede especificarse un rango utilizando un guión (-).

Si la cadena empieza por `^`, no se acepta ninguno de los caracteres posteriores a `^`. Si la cadena no empieza por `^`, se aceptan los caracteres de la cadena. También es posible utilizar `^` para alternar entre los caracteres que se aceptan y los que no se aceptan.

Por ejemplo, en el código siguiente se permite A-Z excepto X y Q:

```
Ta.restrict = "A-Z^XQ";
```

Puede utilizar la barra diagonal inversa (`\`) para introducir un guión (`-`), el símbolo de intercalación (`^`) o el carácter de barra diagonal inversa (`\`), como se muestra a continuación:

```
\^
\-
\\
```

Cuando se introduce el carácter `\` en el panel Acciones entre comillas dobles, tiene un significado especial para el intérprete de comillas dobles del panel Acciones. Significa que el carácter que va a continuación de `\` debe interpretarse literalmente. Por ejemplo, se podría utilizar el siguiente código para introducir una comilla sencilla:

```
var leftQuote = "\'";
```

El intérprete restrict del panel Acciones también utiliza `\` como carácter de escape. Por tanto, es posible que crea que la siguiente expresión funcionará correctamente:

```
myText.restrict = "0-9\-\^\\";
```

Sin embargo, como esta expresión se encuentra entre comillas dobles, el valor siguiente se enviará al intérprete restrict: `0-9-^\` y el intérprete restrict no comprenderá este valor.

Debe escribir esta expresión entre comillas dobles; no sólo debe enviarla al intérprete restrict, sino que también debe escribir una barra diagonal inversa antes de las comillas dobles en el intérprete incorporado del panel Acciones, para interpretarlas literalmente. Para enviar el valor `0-9\-\^\\` al intérprete restrict, debe introducir el siguiente código:

```
myText.restrict = "0-9\\-\\^\\\\";
```

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación Incorporar contornos de fuentes del inspector de propiedades.

Ejemplo

En el siguiente ejemplo se muestran tres usos diferentes de la propiedad `restrict`. El primer uso restringe las entradas a caracteres en mayúsculas de la A a la Z, espacios y números. El segundo uso permite cualquier carácter excepto las minúsculas de la a a la z. El tercer uso permite sólo números, -, ^ y \.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1, utilizando sólo una de las siguientes sentencias `restrict` cada vez.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

// Ejemplo 1: Permitir sólo mayúsculas de la A a la Z, espacios y dígitos
  del 0 al 9.
my_ti.restrict = "A-Z 0-9";

// Ejemplo 2: Permitir todo EXCEPTO minúsculas de la a a la z.
my_ti.restrict = "^a-z";

// Ejemplo 3: Permitir sólo dígitos del 0 al 9, guión (-), ^ y \
my_ti.restrict = "0-9\\-\\^\\\\";
```

TextInput.text

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

textInputInstance.text

Descripción

Propiedad; contenido de texto de un componente TextInput. El valor predeterminado es "" (una cadena vacía).

Ejemplo

El código siguiente coloca una cadena en la instancia de TextInput denominada my_ti y luego rastrea dicha cadena en el panel Salida.

En primer lugar, debe arrastrar un componente TextInput al escenario y asignarle el nombre de instancia `my_ti`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Instancia de TextInput en el escenario (nombre de instancia: my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.text = "The Royal Nonesuch";
trace(my_ti.text); // "The Royal Nonesuch"
```

CAPÍTULO 47

Interfaz TransferObject

47

Nombre de clase de ActionScript mx.data.to.TransferObject

La interfaz TransferObject define un conjunto de métodos que deben implementar los elementos administrados por el componente DataSet. La propiedad `DataSet.itemClassName` especifica el nombre de la clase del objeto de transferencia del que se crea una instancia cada vez que se necesita un elemento nuevo. También puede especificar esta propiedad mediante el inspector de propiedades para un componente DataSet determinado.

Resumen de métodos de la interfaz TransferObject

En la tabla siguiente se enumeran los métodos de la interfaz TransferObject.

Método	Descripción
`TransferObject.clone()`	Crea una instancia del objeto de transferencia.
`TransferObject.getPropertyData()`	Devuelve los datos para este objeto de transferencia.
`TransferObject.setPropertyData()`	Establece los datos para este objeto de transferencia.

TransferObject.clone()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

```
class itemClass implements mx.data.to.TransferObject {
  function clone() {
    // El código se escribe aquí.
  }
}
```

Parámetros

Ninguno.

Valor devuelto

Una copia del objeto de transferencia.

Descripción

Método; crea una instancia del objeto de transferencia. La implementación de este método crea una copia del objeto de transferencia existente y de sus propiedades y, a continuación, devuelve dicho objeto.

Ejemplo

La siguiente función devuelve una copia de este objeto de transferencia con todas las propiedades establecidas con los mismos valores que el original:

```
class itemClass implements mx.data.to.TransferObject {
  function clone():Object {
    var copy:itemClass = new itemClass();
    for (var p in this) {
      copy[p]= this[p];
    }
    return(copy);
  }
}
```

TransferObject.getPropertyData()

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

```
class itemClass implements mx.data.to.TransferObject {
  function getPropertyData() {
  // El código se escribe aquí.
  }
}
```

Parámetros

Ninguno.

Valor devuelto

Un objeto.

Descripción

Método; devuelve los datos para este objeto de transferencia. La implementación de este método puede devolver un objeto ActionScript anónimo con las propiedades y los valores correspondientes.

Ejemplo

La siguiente función devuelve un objeto denominado `internalData` que contiene las propiedades y valores correspondientes del objeto Contact:

```
class Contact implements mx.data.to.TransferObject {
function getPropertyData():Object {
  var internalData:Object = {name:name, readOnly:_readOnly, phone:phone,
  zip:zip.zipPlus4};
  return(internalData);
}
```

TransferObject.setPropertyData()

Disponibilidad

Flash Player 7.

Edición

Flash MX 2004.

Utilización

```
class yourClass implements TransferObject {
  function setPropertyData(propData) {
  // El código se escribe aquí.
  }
}
```

Parámetros

propData Objeto que contiene los datos asignados a este objeto de transferencia.

Valor devuelto

Ninguno.

Descripción

Método; establece los datos para este objeto de transferencia. El parámetro *propData* es un objeto cuyos campos contienen los datos asignados por el componente DataSet a este objeto de transferencia.

Ejemplo

La siguiente función recibe un parámetro *propData* y aplica los valores de sus propiedades a las propiedades del objeto Contact:

```
class Contact implements mx.data.to.TransferObject {

  function setPropertyData(propData: Object):Void {
    _readOnly = propData.readOnly;
    phone = propData.phone;
    zip = new mx.data.types.ZipCode(data.zip);
  }

  public var name:String;
  public var phone:String;
  public var zip:ZipCode;
  private var _readOnly:Boolean; // indica si es inmutable
}
```

CAPÍTULO 48

Clase TransitionManager

48

Nombre de clase de ActionScript mx.transitions.TransitionManager

La clase TransitionManager y las clases basadas en transiciones que definen efectos permiten aplicar rápidamente efectos de animación de transición de gran impacto visual a diapositivas y clips de película.

Como su nombre indica, la clase TransitionManager administra transiciones. Permite aplicar uno de los diez efectos de animación a las diapositivas y clips de película. Al crear componentes personalizados de la versión 2 de la arquitectura de componentes de Macromedia puede utilizar TransitionManager para aplicar efectos de animación a clips de película en la interfaz visual del componente. Los efectos de transición se definen en un conjunto de clases de transición que amplían la clase base `mx.transitions.Transition`. Las transiciones sólo se aplican a través de una instancia de TransitionManager; no se crean instancias de transiciones directamente. La clase TransitionManager implementa eventos de animación.

Utilización de la clase TransitionManager

Para utilizar los métodos y las propiedades de la clase TransitionManager, tiene dos opciones para crear una nueva instancia. La más sencilla consiste en llamar al método `TransitionManager.start()`, que crea una nueva instancia de TransitionManager, designa el objeto de destino, aplica una transición con un método de suavizado e inicia todo en una sola llamada. El código siguiente utiliza el método `TransitionManager.start()`:

```
mx.transitions.TransitionManager.start(myMovieClip_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

Para más información sobre el método `TransitionManager.start()`, su uso y sus parámetros, consulte `TransitionManager.start()` en la página 1283.

También puede crear una nueva instancia de la clase TransitionManager mediante el operador new. A continuación debe designar las propiedades de transición e iniciar el efecto de transición en un segundo paso llamando al método `TransitionManager.startTransition()`. El código siguiente utiliza el método `TransitionManager.startTransition()`:

```
var myTransitionManager:mx.transitions.TransitionManager = new
  mx.transitions.TransitionManager(myMovieClip_mc);
myTransitionManager.startTransition({type:mx.transitions.Zoom,
  direction:Transition.IN, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut});
```

Parámetros de la clase TransitionManager

Cuando cree una nueva instancia de una clase TransitionManager mediante el operador *new*, debe designar un clip de película de destino en el parámetro `content` para su constructor. El constructor de la clase mx.transitions.TransitionManager tiene el siguiente nombre y tipo de parámetro:

```
TransitionManager(content:MovieClip)
```

content es el objeto de clip de película al que la instancia de TransitionManager aplica una transición.

NOTA	Si crea una instancia de TransitionManager utilizando el operador new, debe designar a continuación las propiedades de la transición que desea aplicar y hacer una llamada para iniciar la transición mediante el método `TransitionManager.startTransition()`; de lo contrario, la transición no se aplica a un clip de película o no se inicia. Para ver más detalles sobre el método `TransitionManager.startTransition()`, su uso y sus parámetros, consulte`TransitionManager.startTransition()` en la página 1284. Una alternativa rápida al proceso en dos pasos para crear una instancia de TransitionManager es llamar simplemente al método `TransitionManager.start()`; para más información, consulte`TransitionManager.start()` en la página 1283. El método `TransitionManager.start()` permite crear una instancia de TransitionManager, proporcionar el clip de película de destino y especificar las propiedades de transición en una llamada.

Especificación de un método y una clase de suavizado en una transición

Al crear una instancia de la clase TransitionManager mediante el método `TransitionManager.start()`, utiliza la propiedad easing del parámetro `transParam` para especificar una función o un método que proporciona un cálculo de suavizado. Para ver una descripción completa de los métodos y las clases de suavizado disponibles, consulte "Especificación de un método y una clase de suavizado en una transición" en la página 1277.

Resumen de la clase TransitionManager

En las siguientes secciones se muestran los métodos, las propiedades y los eventos de la clase TransitionManager.

Resumen de métodos de la clase TransitionManager

En la tabla siguiente se enumeran los métodos de la clase TransitionManager.

Método	Descripción
`TransitionManager.start()`	Crea una nueva instancia de TransitionManager, designa el objeto de destino, aplica una transición y la inicia.
`TransitionManager.startTransition()`	Crea una instancia de la transición y la inicia.
`TransitionManager.toString()`	Devuelve el tipo de TransitionManager como una cadena.

Resumen de propiedades de la clase TransitionManager

En la tabla siguiente se enumeran las propiedades de la clase TransitionManager.

Propiedad	Descripción
`TransitionManager.content`	La instancia de clip de película a la que TransitionManager va a aplicar una transición.
`TransitionManager.contentAppearance`	Objeto que contiene las propiedades visuales guardadas del contenido (clip de película de destino) en el que se aplicarán las transiciones. Es una propiedad de sólo lectura.

Resumen de eventos de la clase TransitionManager

En la tabla siguiente se enumeran los eventos de la clase TransitionManager.

Evento	Descripción
`TransitionManager.allTransitionsInDone`	Lo difunde una instancia de TransitionManager cuando completa una transición con la propiedad direction establecida en `mx.transitions.Transition.IN`.
`TransitionManager.allTransitionsOutDone`	Lo difunde una instancia de TransitionManager cuando completa una transición con la propiedad direction establecida en `mx.transitions.Transition.OUT`.

TransitionManager.allTransitionsInDone

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.allTransitionsInDone = function(eventObj:Object) {
  // ...
};
transitionManagerInstance.addEventListener("allTransitionsInDone",
  listenerObject);
```

Descripción

Evento; notifica a los detectores que la instancia de TransitionManager ha completado todas las transiciones que tienen la propiedad direction establecida en `mx.transitions.Transition.IN` y las ha eliminado de la lista de transiciones que deben aplicarse.

El ejemplo de sintaxis utiliza un modelo de eventos de distribuidor o detector. Una instancia de TransitionManager (*transitionManagerInstance*) distribuye un evento (en este caso, allTransitionsInDone) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (listenerObject) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (eventObject) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. En este caso, el evento allTransitionsInDone proporciona una propiedad de destino que contiene la instancia de TransitionManager que activó el evento, lo que permite utilizar la instancia y todas sus propiedades y métodos en el código que recibe el evento allTransitionsInDone. Para más información, consulte Capítulo 21, “Clase EventDispatcher”, en la página 515.

Ejemplo

En el siguiente ejemplo de código se asigna un objeto para que detecte el evento allTransitionsInDone y se especifica el método que actuará como controlador del evento. Cuando se llama a ese método para controlar el evento, ya se ha completado una transición con la propiedad direction establecida en mx.transitions.Transition.IN.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Iris, direction:Transition.IN,
  duration:1, easing:None.easeNone, startPoint:5, shape:Iris.CIRCLE});

var myListener:Object = new Object();
myListener.allTransitionsInDone = function(eventObj:Object) {
  trace("allTransitionsInDone event occurred.");
};
myTransitionManager.addEventListener("allTransitionsInDone", myListener);
```

Véase también

Capítulo 21, “Clase EventDispatcher”, en la página 515

TransitionManager.allTransitionsOutDone

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.allTransitionsOutDone = function(eventObj:Object) {
  // ...
};
transitionManagerInstance.addEventListener("allTransitionsOutDone",
  listenerObject);
```

Descripción

Evento; notifica a los detectores que la instancia de TransitionManager ha completado todas las transiciones que incluyen una propiedad de dirección "out" y las ha eliminado de la lista de transiciones que deben aplicarse.

El ejemplo de sintaxis utiliza un modelo de eventos de distribuidor o detector. Una instancia de TransitionManager (`transitionManagerInstance`) distribuye un evento (en este caso, `allTransitionsOutDone`) y éste se controla mediante una función, también denominada controlador, asociada con el objeto detector (`listenerObject`) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. En este caso, el evento `allTransitionsInDone` proporciona una propiedad de destino que contiene la instancia de TransitionManager que activó el evento, lo que permite utilizar la instancia y todas sus propiedades y métodos en el código que recibe el evento `allTransitionsInDone`. Para más información, consulte Capítulo 21, "Clase EventDispatcher", en la página 515.

Ejemplo

En el siguiente ejemplo de código se asigna un objeto para que detecte el evento `allTransitionsOutDone` y se especifica el método que actuará como controlador del evento. Cuando se llama a ese método para controlar el evento, ya se ha completado una transición con la propiedad direction establecida en `mx.transitions.Transition.IN`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Iris, direction:Transition.OUT,
  duration:1, easing:None.easeNone,startPoint:5, shape:Iris.CIRCLE});

var myListener:Object = new Object();
myListener.allTransitionsOutDone = function(eventObj:Object) {
  trace("allTransitionsOutDone event occurred.");
};
myTransitionManager.addEventListener("allTransitionsOutDone", myListener);
```

Véase también

Capítulo 21, “Clase EventDispatcher”, en la página 515

TransitionManager.content

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`transitionManagerInstance``.content`

Descripción

Propiedad; la instancia de clip de película a la que TransitionManager va a aplicar una transición.

Ejemplo

En el siguiente ejemplo se devuelve el objeto de clip de película en el que actualmente se aplica la instancia de TransitionManager:

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
var myMovieClip:MovieClip = myTransitionManager.content;
trace(myMovieClip._name);
```

TransitionManager.contentAppearance

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

transitionManagerInstance.contentAppearance

Descripción

Propiedad (sólo lectura); objeto que contiene una instantánea de las propiedades del clip de película de destino de una instancia de TransitionManager antes de que se aplique la transición. Este objeto resulta útil para obtener información sobre los valores de propiedad que se puede esperar que el clip de película devuelva cuando la transición haya finalizado. El objeto devuelto por `TransitionManager.contentAppearance` contiene un registro de los valores originales correspondientes de los clips de película de destino en las siguientes propiedades: `_x`, `_y`, `_xscale`, `_yscale`, `_alpha`, `_rotation`, `_innerBounds`, `_outerBounds`, `_width`, `_height` y `colorTransform`. Estas propiedades se guardan y se llama al método `TransitionManager.start()` o `TransitionManager.startTransition()`.

Ejemplo

El ejemplo siguiente llama a `TransitionManager.contentAppearance()` para obtener la configuración original de las propiedades del clip de película de destino del objeto TransitionManager antes de que se aplique la transición:

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Zoom, direction:Transition.OUT,
  duration:3, easing:Bounce.easeOut});

var myMovieClip:MovieClip = myTransitionManager.content;
var myOriginalMovieClipProps:Object =
  myTransitionManager.contentAppearance;

for (var prop in myOriginalMovieClipProps) {
 trace(myMovieClip._name + "." +prop+" = "+myOriginalMovieClipProps[prop]);
}
```

TransitionManager.start()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

transitionManagerInstance.start(*content*, *transParams*)

Parámetros

content El objeto MovieClip al que se va aplicar el efecto de transición.

transParams Colección de parámetros que se pasan en un objeto.

El objeto `transParams` debe contener un parámetro `type` que indica la clase de efecto de transición que se va a aplicar, seguido de los parámetros `direction`, `duration` y `easing`. Además, debe incluir los parámetros requeridos por la clase de efecto de la transición. Por ejemplo, la clase de efecto de la transición `mx.transitions.Iris` requiere los parámetros adicionales `startPoint` y `shape`. Así, además de los parámetros `type`, `duration` y `easing` que toda transición requiere, también debe añadir (al objeto `transParams`) los parámetros `startPoint` y `shape` que requiere el efecto `mx.transitions.Iris`. El código siguiente añade parámetros `startPoint` y `shape` al efecto `mx.transitions.Iris`:

```
{type:mx.transitions.Iris, direction:mx.transitions.Transition.IN,
  duration:5, easing:mx.transitions.easing.Bounce.easeOut,
  startPoint:5, shape:mx.transitions.Iris.CIRCLE}
```

Para comprobar los parámetros requeridos adicionales para el efecto de la clase de transición que está especificando en el parámetro `type` del objeto `transParam`, consulte la API de esa clase de transición. Por ejemplo, para más información sobre la clase de transición Blinds, consulte"Transición Blinds" en la página 1288.

NOTA

El parámetro type del objeto `transParams` debe incluir el nombre completo del paquete de clase de las clases especificadas para sus parámetros, a menos que ya hayan sido importadas mediante la sentencia `import`. Para no tener que especificar el nombre completo del paquete de clase para toda la colección de parámetros `transParams`, debe colocar previamente las siguientes instrucciones `import` en el código para importar todas las clases `mx.transitions` y todas las clases `mx.transitions.easing`:

```
import mx.transitions.*;
import mx.transitions.easing.*;
```

Valor devuelto

Una instancia del objeto Transition en el que debe aplicarse la instancia de TransitionManager.

Descripción

Método; crea una instancia de la clase TransitionManager si no existe ninguna, crea una instancia de la clase de transición especificada designada en el parámetro transParams.type y, a continuación, inicia la transición. La transición se aplica a la diapositiva o el clip de película designado en el parámetro content.

Ejemplo

El código siguiente utiliza el método `TransitionManager.start()` para crear una instancia de TransitionManager y asigna una transición Iris a un clip de película denominado img1_mc. El método `TransitionManager.start()` contiene dos parámetros. El primer parámetro, `content`, es el objeto MovieClip al que se aplicará el efecto de la transición. El segundo parámetro para el método `TransitionManager.start()`, `transParam`, contiene un objeto en el que se almacena una colección de parámetros. Este objeto que contiene una colección de parámetros designa primero el tipo de efecto de transición con el parámetro `type`, seguido de los parámetros `direction`, `duration` y `easing`. Los parámetros `type`, `direction`, `duration` y `easing` son datos necesarios para todos los efectos de TransitionManager. A continuación del parámetro `easing` van los parámetros que el tipo de transición requiere específicamente. En el siguiente ejemplo, el tipo de transición es Iris, que requiere los parámetros `startPoint` y `shape` (para obtener más información sobre los parámetros de la transición Iris, consulte "Transición Iris" en la página 1290):

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Iris, direction:Transition.IN,
  duration:5, easing:Bounce.easeOut, startPoint:5, shape:Iris.CIRCLE});
```

TransitionManager.startTransition()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`transitionManagerInstance``.startTransition(`*`transParams`*`)`

Parámetros

transParams Colección de parámetros que se pasan en un objeto.

El objeto `transParams` debe contener un parámetro `type` que indica la clase de efecto de transición que se va a aplicar, seguido de los parámetros `direction`, `duration` y `easing`. Además, debe incluir los parámetros requeridos por la clase de efecto de transición especificada. Por ejemplo, la clase de efecto de la transición `mx.transitions.Iris` requiere los parámetros adicionales `startPoint` y `shape`. Así, además de los parámetros `type`, `duration` y `easing` que toda transición requiere, también debe añadir (al objeto `transParams`) los parámetros `startPoint` y `shape` que requiere el efecto `mx.transitions.Iris`. El código siguiente añade parámetros `startPoint` y `shape` al efecto `mx.transitions.Iris`:

```
{type:mx.transitions.Iris, direction:mx.transitions.Transition.IN,
  duration:5, easing:mx.transitions.easing.Bounce.easeOut, startPoint:5,
  shape:mx.transitions.Iris.CIRCLE}
```

Para comprobar los parámetros requeridos adicionales para el efecto de la clase de transición que está especificando en el parámetro type del objeto `transParam`, consulte la API de esa clase de transición. Por ejemplo, para más información sobre la clase de transición Blinds, consulte"Transición Blinds" en la página 1288.

NOTA

El parámetro type del objeto `transParams` debe incluir el nombre completo del paquete de clase de las clases especificadas para sus parámetros, a menos que ya hayan sido importadas mediante la sentencia `import`. Para no tener que especificar el nombre completo del paquete de clase para toda la colección de parámetros `transParams`, debe colocar previamente las siguientes instrucciones `import` en el código para importar todas las clases `mx.transitions` y todas las clases `mx.transitions.easing`:

```
import mx.transitions.*;
import mx.transitions.easing.*;
```

Valor devuelto

Una instancia del objeto Transition en el que debe aplicarse la instancia de TransitionManager.

Descripción

Método; crea e inicia una instancia de la clase de transición especificada que se aplicará a la diapositiva o al clip de película que se verá afectado por la instancia de TransitionManager. Si ya hay una transición coincidente, dicha transición se elimina y se crea e inicia una nueva.

Ejemplo

El código siguiente importa la clase TransitionManager y crea una nueva instancia de TransitionManager. A continuación, el método `TransitionManager.startTransition()` designa una transición `mx.transitions.Zoom` en el parámetro `type`. El parámetro `direction` debe establecerse en `mx.transitions.Transition.OUT` para indicar que la transición debe moverse en el sentido saliente. La duración de la transición es 3 segundos. El suavizado se calcula mediante el método `mx.transitions.Bounce.easeOut()` de la clase Bounce. Este efecto hace que el clip de película img1_mc parezca alejarse en un movimiento de rebote hasta que desaparece; la duración del efecto completo es de 3 segundos.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Zoom, direction:Transition.OUT,
  duration:3, easing:Bounce.easeOut});
```

TransitionManager.toString()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

transitionManagerInstance.toString()

Valor devuelto

Se devuelve la siguiente cadena: `"[TransitionManager]"`.

Descripción

Método; devuelve el tipo del objeto TransitionManager como una cadena.

Ejemplo

El siguiente código indica a la instancia de TransitionManager que devuelva una cadena donde se especifique su tipo:

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
var myType:String = myTransitionManager.toString();
trace(myType);
```

Clases basadas en transiciones

Herencia (clase raíz)

Nombre de clase de ActionScript mx.transitions.Transition

La clase Transition es la clase base de todas las clases de transición. No es posible utilizar esta clase ni acceder a ella directamente. Permite que las clases basadas en transiciones compartan algunos comportamientos y propiedades comunes a los que se accede a través de una instancia de la clase TransitionManager. Las clases basadas en Transition definen un efecto que se aplica a lo largo del tiempo a una diapositiva o clip de película.

Flash incluye diez transiciones que puede utilizar para aplicar efectos a objetos de clip de película. Puede personalizar todas las transiciones incluyendo métodos de suavizado opcionales; la mayoría de las transiciones admiten varios parámetros opcionales que permiten controlar aspectos concretos de su efecto. El *suavizado* es la aceleración o desaceleración gradual durante una animación, que hace que las animaciones parezcan más realistas. Por ejemplo, una bola puede aumentar gradualmente su velocidad hacia el comienzo de una animación e ir reduciendo la velocidad antes de detenerse por completo al finalizar la animación. Existen muchas ecuaciones para esta aceleración y desaceleración que cambian la animación de suavizado.

Las transiciones se usan con la clase TransitionManager. Véase "Clase TransitionManager" en la página 1275. La clase TransitionManager se utiliza para especificar una transición y aplicarla a un objeto de clip de película en lugar de llamarla directamente. Por ejemplo, para aplicar una transición Zoom a un clip de película denominado img1_mc, debe especificar la clase de transición Zoom como el parámetro type de `TransitionManager.start()`:

```
mx.transitions.TransitionManager.start(myMovieClip_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

Flash incluye las siguientes transiciones:

Transición	Descripción
Transición Blinds	Muestra el objeto de clip de película mediante rectángulos que aparecen o desaparecen.
Transición Fade	Hace que el clip de película se desvanezca y vuelva a aparecer.
Transición Fly	Desliza el objeto de clip de película desde una dirección especificada.
Transición Iris	Muestra u oculta el objeto de clip de película mediante una máscara animada de forma circular o cuadrada que se acerca o aleja.
Transición Photo	Hace que el objeto de clip de película aparezca o desaparezca como un flash fotográfico.
Transición PixelDissolve	Muestra u oculta el objeto de clip de película mediante rectángulos que aparecen o desaparecen aleatoriamente en un patrón de tablero de ajedrez.
Transición Rotate	Gira el objeto de clip de película.
Transición Squeeze	Ajusta el tamaño del objeto de clip de película horizontal o verticalmente.
Transición Wipe	Muestra u oculta el objeto de clip de película mediante una máscara animada que se mueve horizontalmente.
Transición Zoom	Acerca o aleja el objeto de clip de película y ajusta su proporción.

NOTA

Las transiciones sólo están disponibles en ActionScript 2.0.

Transición Blinds

Nombre de clase de ActionScript mx.transitions.Blinds

Parámetros

numStrips Número de bandas de máscara en el efecto de persiana. El rango recomendado es de 1 a 50.

dimension Entero que indica que las bandas de Blinds deben ser verticales (0) u horizontales (1).

Descripción

Efecto de transición: muestra el objeto de clip de película mediante rectángulos que aparecen o desaparecen.

Esta clase se utiliza especificando `mx.transitions.Blinds` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Blinds con diez `numStrips` y un entero `dimension` especificado como vertical (0). El destino de contenido de la transición es el clip de película img1_mc. La instancia de TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos sin suavizado.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Blinds, direction:Transition.IN,
  duration:2, easing:None.easeNone, numStrips:10, dimension:0});
```

Transición Fade

Nombre de clase de ActionScript mx.transitions.Fade

Descripción

Efecto de transición: hace que el clip de película se desvanezca y vuelva a aparecer.

Esta clase se utiliza especificando `mx.transitions.Fade` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Fade. El destino de contenido de la transición es el clip de película img1_mc. La instancia de TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 3 segundos sin suavizado.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Fade, direction:Transition.IN,
  duration:3, easing:None.easeNone});
```

Transición Fly

Nombre de clase de ActionScript mx.transitions.Fly

Parámetros

`startPoint` Entero que indica una posición inicial; el intervalo es de 1 a 9:

superior izquierdo, 1; superior central, 2; superior derecho, 3; izquierdo central, 4; central, 5; derecho central, 6; inferior izquierdo, 7; inferior central, 8; inferior derecho, 9.

Descripción

Efecto de transición: desliza el objeto de clip de película desde una dirección especificada.

Esta clase se utiliza especificando `mx.transitions.Fly` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El código siguiente crea una instancia de TransitionManager que aplica la transición Fly con `startPoint` establecido en la parte inferior derecha (9). El destino de contenido de la transición es el clip de película img1_mc. La instancia de TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 3 segundos con un efecto de suavizado `Elastic.easeOut`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Fly, direction:Transition.IN,
  duration:3, easing:Elastic.easeOut, startPoint:9});
```

Transición Iris

Nombre de clase de ActionScript mx.transitions.Iris

Parámetros

`startPoint` Entero que indica una posición inicial; el intervalo es de 1 a 9:

superior izquierdo, 1; superior central: 2; superior derecho, 3; izquierdo central, 4; central, 5; derecho central, 6; inferior izquierdo, 7; inferior central, 8; inferior derecho, 9.

`shape` Forma de máscara de `mx.transitions.Iris.SQUARE` (un cuadrado) o `mx.transitions.Iris.CIRCLE` (un círculo).

Descripción

Efecto de transición: Muestra el objeto de clip de película mediante una máscara animada de forma circular o cuadrada que se acerca o aleja.

Esta clase se utiliza especificando `mx.transitions.Iris` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Iris con `startPoint` desde el centro (5) y con la forma de máscara `mx.transitions.Iris.CIRCLE`. El destino de `content` de la transición es el clip de película img1_mc. TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos con suavizado Strong y énfasis en easeOut, especificando el método de cálculo de suavizado `mx.transitions.easing.Strong.easeOut`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Iris, direction:Transition.IN,
  duration:2, easing:Strong.easeOut, startPoint:5, shape:Iris.CIRCLE});
```

Transición Photo

Nombre de clase de ActionScript mx.transitions.Photo

Descripción

Efecto de transición: hace que el objeto de clip de película aparezca o desaparezca como un flash fotográfico.

Esta clase se utiliza especificando `mx.transitions.Photo` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Photo al destino de contenido: el clip de película img1_mc. La clase TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 1 segundo sin suavizado.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start (img1_mc, {type:Photo, direction:Transition.IN,
  duration:1, easing:None.easeNone});
```

Transición PixelDissolve

Nombre de clase de ActionScript mx.transitions.PixelDissolve

Parámetros

`xSections` Entero que indica el número de secciones rectangulares de máscara a lo largo del eje horizontal. El rango recomendado es de 1 a 50.

`ySections` Entero que indica el número de secciones rectangulares de máscara a lo largo del eje vertical. El rango recomendado es de 1 a 50.

Descripción

Efecto de transición: Muestra el objeto de clip de película mediante rectángulos que aparecen o desaparecen aleatoriamente formando un patrón de tablero de ajedrez.

Esta clase se utiliza especificando `mx.transitions.PixelDissolve` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición PixelDissolve con diez `xSections` y diez `ySections`. El destino de contenido de la transición es el clip de película img1_mc. La instancia de TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos sin suavizado.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:PixelDissolve,
  direction:Transition.IN, duration:2, easing:None.easeNone, xSections:10,
  ySections:10});
```

Transición Rotate

Nombre de clase de ActionScript mx.transitions.Rotate

Parámetros

`ccw` Valor booleano: *`false`* si se gira en el sentido de las agujas del reloj; *`true`* si se gira en el sentido contrario.

`degrees` Entero que indica el número de grados que debe girar el objeto. El rango recomendado es de 1 a 9999. Por ejemplo, si se establece `degrees` en 1080, el objeto giraría completamente tres veces.

Descripción

Efecto de transición: gira el objeto de clip de película.

Esta clase se utiliza especificando `mx.transitions.Rotate` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplicará la transición Rotate 720 grados (degrees) en el sentido de las agujas del reloj (dos giros completos). El destino de contenido de la transición es el clip de película img1_mc. La instancia de TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 3 segundos con el suavizado establecido en `Strong.easeInOut`, de forma que la transición comienza despacio, se acelera y después finaliza despacio.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Rotate, direction:Transition.IN,
  duration:3, easing:Strong.easeInOut, ccw:false, degrees:720});
```

Transición Squeeze

Nombre de clase de ActionScript mx.transitions.Squeeze

Parámetros

dimension Entero que indica que el efecto Squeeze debe ser horizontal (0) o vertical (1).

Descripción

Efecto de transición: ajusta el tamaño del objeto de clip de película horizontal o verticalmente.

Esta clase se utiliza especificando `mx.transitions.Squeeze` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Squeeze con un entero `dimension` especificado como vertical (1). El destino de contenido de la transición es el clip de película img1_mc. TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos con un efecto de suavizado Elastic en el sentido easeOut.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Squeeze, direction:Transition.IN,
  duration:2, easing:Elastic.easeOut, dimension:1});
```

Transición Wipe

Nombre de clase de ActionScript mx.transitions.Wipe

Parámetros

startPoint Entero que indica una posición inicial. Intervalo de 1 a 4 y de 6 a 9:

superior izquierdo, 1; superior central, 2; superior derecho, 3; izquierdo central, 4; derecho central, 6; inferior izquierdo, 7; inferior central, 8; inferior derecho, 9.

Descripción

Efecto de transición: muestra u oculta el objeto de clip de película mediante una máscara animada que se mueve horizontalmente.

Esta clase se utiliza especificando `mx.transitions.Wipe` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Wipe con `startPoint` desde la parte superior izquierda (1). El destino de contenido de la transición es el clip de película img1_mc. TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos sin suavizado.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Wipe, direction:Transition.IN,
  duration:2, easing:None.easeNone, startPoint:1});
```

Transición Zoom

Nombre de clase de ActionScript mx.transitions.Zoom

Descripción

Efecto de transición: acerca o aleja el objeto de clip de película y ajusta su proporción.

Esta clase se utiliza especificando `mx.transitions.Zoom` como un parámetro `transObject.type` para la clase TransitionManager.

Ejemplo

El siguiente código crea una instancia de TransitionManager que aplica la transición Zoom al destino de contenido: el clip de película img1_mc. TransitionManager aplica un sentido `mx.transitions.Transition.IN` durante 2 segundos con un suavizado de tipo Elastic que se inicia rápidamente y se suaviza lentamente al final.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Zoom, direction:Transition.IN,
  duration:2, easing:Elastic.easeOut});
```

CAPÍTULO 49

Interfaz TreeDataProvider (sólo en Flash Professional)

49

La interfaz TreeDataProvider es un conjunto de propiedades y métodos, y no es necesario crear una instancia de la misma para poder utilizarla. Si una clase Tree se empaqueta en un archivo SWF, todas las instancias XML del archivo SWF contienen la interfaz TreeDataProvider. Todos los nodos de un árbol son objetos XML que contienen la interfaz TreeDataProvider.

Es recomendable utilizar los métodos de TreeDataProvider para crear XML para la propiedad `Tree.dataProvider`, ya que sólo TreeDataProvider difunde eventos que actualizan la visualización del árbol. Son eventos gestionados por la clase Tree; no es necesario escribir funciones para gestionar estos eventos. Los métodos de la clase XML incorporados no difunden dichos eventos.

Se pueden utilizar los métodos de TreeDataProvider para controlar el modelo de datos y la visualización de datos. Se pueden utilizar los métodos de la clase XML incorporados para realizar tareas de sólo lectura como la navegación por la jerarquía del árbol.

La propiedad que contiene el texto que debe mostrarse puede seleccionarse especificando la propiedad `labelField` o `labelFunction`. Por ejemplo, `myTree.labelField = "firstName";` indica que se consulte la propiedad `myTreeDP.attributes.fred` para obtener el texto que se va a visualizar.

Resumen de métodos de la interfaz TreeDataProvider

En la tabla siguiente se enumeran los métodos de la interfaz TreeDataProvider.

Método	Descripción
`TreeDataProvider.addTreeNode()`	Añade un nodo secundario a la raíz del árbol.
`TreeDataProvider.addTreeNodeAt()`	Añade un nodo secundario en una ubicación especificada del nodo principal.

Método	Descripción
TreeDataProvider.getTreeNodeAt()	Devuelve el nodo secundario especificado de un nodo.
TreeDataProvider.removeTreeNode()	Elimina un nodo y todos sus descendientes del nodo principal de dicho nodo.
TreeDataProvider.removeTreeNodeAt()	Elimina un nodo y todos sus descendientes de la posición de índice del nodo secundario.

Resumen de propiedades de la interfaz TreeDataProvider

En la tabla siguiente se enumeran las propiedades de la interfaz TreeDataProvider.

Propiedad	Descripción
TreeDataProvider.attributes.data	Especifica los datos que van a asociarse con un nodo.
TreeDataProvider.attributes.label	Especifica el texto que va a mostrarse junto a un nodo.

TreeDataProvider.addTreeNode()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
someNode.addTreeNode(label, data)
```

Sintaxis 2:

```
someNode.addTreeNode(child)
```

Parámetros

label Cadena que muestra el nodo.

data Objeto de cualquier tipo que está asociado con el nodo.

child Cualquier objeto XMLNode.

Valor devuelto

El nodo XML añadido.

Descripción

Método; añade un nodo secundario a la raíz del árbol. El nodo se crea a partir de la información proporcionada en los parámetros *label* y *data* (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto XMLNode (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

La llamada a este método actualiza la visualización del árbol.

Ejemplo

La primera línea de código del ejemplo siguiente localiza el nodo al que debe añadirse un nodo secundario. La segunda línea añade un nuevo nodo a un nodo especificado.

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.addTreeNode("Inbox", 3);
```

El código siguiente traslada un nodo de un árbol a la raíz de otro árbol:

```
myTreeNode.addTreeNode(mySecondTree.getTreeNodeAt(3));
```

TreeDataProvider.addTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
someNode.addTreeNodeAt(index, label, data)
```

Sintaxis 2:

```
someNode.addTreeNodeAt(index, child)
```

Parámetros

index Entero que indica la posición de índice (entre los nodos secundarios) en la que debe añadirse el nodo.

label Cadena que muestra el nodo.

data Objeto de cualquier tipo que está asociado con el nodo.

child Cualquier objeto XMLNode.

Valor devuelto

El nodo XML añadido.

Descripción

Método; añade un nodo secundario a la ubicación especificada del nodo principal. El nodo se crea a partir de la información proporcionada en los parámetros *label* y *data* (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto XMLNode (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

La llamada a este método actualiza la visualización del árbol.

Ejemplo

El código siguiente localiza el nodo al que se añadirá un nodo y añade un nuevo nodo como segundo nodo secundario de la raíz:

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.addTreeNodeAt(1, "Inbox", 3);
```

El código siguiente desplaza un nodo de un árbol para que sea el cuarto nodo secundario de la raíz de otro árbol:

```
myTreeNode.addTreeNodeAt(3, mySecondTree.getTreeNodeAt(3));
```

TreeDataProvider.attributes.data

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
someNode.attributes.data
```

Descripción

Propiedad; especifica los datos que van a asociarse con el nodo. Esto añade el valor como un atributo del objeto XMLNode. Al definir esta propiedad no se actualizan las visualizaciones del árbol. Esta propiedad puede ser de cualquier tipo de datos.

Ejemplo

El código siguiente localiza el nodo que va a ajustarse y define su propiedad `data`:

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.attributes.data = "hi"; // genera el resultado <node data =
  "hi">;
```

Véase también

`TreeDataProvider.attributes.label`

TreeDataProvider.attributes.label

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`someNode``.attributes.label`

Descripción

Propiedad; cadena que especifica el texto que se visualiza para el nodo. El texto se escribe en un atributo del objeto XMLNode. Al definir esta propiedad no se actualizan las visualizaciones del árbol.

Ejemplo

El código siguiente localiza el nodo que va a ajustarse y define su propiedad `label`.
El resultado del código siguiente es `<node label="Mail">`.

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.attributes.label = "Mail";
```

Véase también

`TreeDataProvider.attributes.data`

TreeDataProvider.getTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
someNode.getTreeNodeAt(index)
```

Parámetros

index Entero que representa la posición del nodo secundario en el nodo actual.

Valor devuelto

El nodo especificado.

Descripción

Método; devuelve el nodo secundario especificado del nodo.

Ejemplo

El código siguiente localiza un nodo y obtiene el segundo nodo secundario de `myTreeNode`:

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.getTreeNodeAt(1);
```

TreeDataProvider.removeTreeNode()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
someNode.removeTreeNode()
```

Parámetros

Ninguno.

Valor devuelto

El nodo XML eliminado o `undefined` si se produce un error.

Descripción

Método; elimina el nodo especificado, y los descendientes, de su nodo principal.

Ejemplo

El código siguiente elimina un nodo:

```
myTreeDP.firstChild.removeTreeNode();
```

TreeDataProvider.removeTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`someNode``.removeTreeNodeAt(`*`index`*`)`

Parámetros

`index` Entero que indica la posición del nodo que debe eliminarse.

Valor devuelto

El nodo XML eliminado o `undefined` si se produce un error.

Descripción

Método; elimina un nodo (y todos sus descendientes) especificado por el nodo actual y la posición de índice del nodo secundario. Al llamar a este método se actualiza la vista.

Ejemplo

El código siguiente elimina el cuarto nodo secundario del nodo `myTreeDP.firstChild`:

```
myTreeDP.firstChild.removeTreeNodeAt(3);
```

CAPÍTULO 50

Componente Tree (sólo en Flash Professional)

50

El componente Tree permite a un usuario visualizar datos jerárquicos. El árbol aparece dentro de un cuadro como el componente List, pero cada elemento de un árbol se denomina *nodo* y puede ser una *hoja* o una *rama*. De forma predeterminada, una hoja se representa con una etiqueta de texto junto a un icono de archivo, mientras que una rama se representa con una etiqueta de texto junto a un icono de carpeta con una flecha de ampliación (triángulo de difusión) que el usuario puede abrir para mostrar los nodos secundarios. Los elementos secundarios de una rama pueden ser hojas o ramas.

Los datos de un componente Tree deben proporcionarse desde un origen de datos XML. Para más información, consulte "Utilización del componente Tree (sólo en Flash Professional)" en la página 1304.

Si se selecciona una instancia Tree al hacer clic en ella o mediante la tabulación, podrá utilizar las teclas siguientes para controlar dicha instancia:

Tecla	Descripción
Flecha abajo	Desplaza la selección un elemento hacia abajo.
Flecha arriba	Desplaza la selección un elemento hacia arriba.
Flecha derecha	Abre un nodo de rama seleccionado. Si una rama ya está abierta, se desplaza al primer nodo secundario.
Flecha izquierda	Cierra un nodo de rama seleccionado. Si se encuentra en un nodo hoja de un nodo de rama cerrado, se desplaza al nodo principal.
Espacio	Abre o cierra un nodo de rama seleccionado.
Fin	Desplaza la selección a la parte inferior de la lista.
Inicio	Desplaza la selección a la parte superior de la lista.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Control	Permite varias selecciones no consecutivas.
Mayús	Permite varias selecciones consecutivas.

Los lectores de pantalla no tienen acceso al componente Tree.

Utilización del componente Tree (sólo en Flash Professional)

El componente Tree se puede utilizar para representar datos jerárquicos como las carpetas del cliente de correo electrónico, los paneles del explorador de archivos o los sistemas de exploración por categorías para el inventario. En muchas ocasiones, los datos de un árbol se recuperan desde un servidor con formato XML, pero también con formato XML bien formado y creado mientras se edita en Flash. El mejor modo de crear XML para el árbol es utilizar la interfaz TreeDataProvider. También puede utilizar la clase XML de ActionScript o crear una cadena XML. Después de crear un origen de datos XML (o cargar uno desde un origen externo), debe asignarlo a `Tree.dataProvider`.

El componente Tree está formado por dos conjuntos de API: la clase Tree y la interfaz TreeDataProvider. La clase Tree contiene los métodos y las propiedades de configuración visual. La interfaz TreeDataProvider permite crear XML y añadirlo a varias instancias de árbol. Un objeto TreeDataProvider difunde los cambios a todos los árboles que lo utilicen. Asimismo, se asignarán los métodos y las propiedades de TreeDataProvider a cualquier objeto XML o XMLNode que se encuentre en el mismo fotograma que un árbol o un menú. Para más información, consulte "Interfaz TreeDataProvider (sólo en Flash Professional)" en la página 1295.

Aplicación de formato XML para el componente Tree

El componente Tree se ha diseñado para mostrar estructuras jerárquicas de datos utilizando XML como modelo de datos. Es importante comprender la relación existente entre el origen de datos XML y el componente Tree.

Observe el siguiente ejemplo de origen de datos XML:

```
<node>
  <node label="Mail">
       <node label="INBOX"/>
       <node label="Personal Folder">
           <node label="Business" isBranch="true" />
           <node label="Demo" isBranch="true" />
           <node label="Personal" isBranch="true" />
           <node label="Saved Mail" isBranch="true" />
           <node label="bar" isBranch="true" />
       </node>
       <node label="Sent" isBranch="true" />
       <node label="Trash"/>
  </node>
</node>
```

NOTA

El atributo `isBranch` es de sólo lectura; no se puede configurar directamente. Para definirlo, llame al método `Tree.setIsBranch()`.

Los nodos del origen de datos XML pueden tener cualquier nombre. Observe en el ejemplo anterior que cada nodo recibe el nombre genérico `node`. El árbol lee el objeto XML y crea la jerarquía de visualización basándose en la relación anidada de los nodos.

Cada uno de los nodos XML se puede visualizar como uno de los dos tipos que hay en el árbol: rama u hoja. Los nodos rama pueden contener varios nodos secundarios y aparecer como un icono de carpeta con una flecha de ampliación, que permite a los usuarios abrir y cerrar la carpeta. Los nodos hoja aparecen como un icono de archivo y no pueden contener nodos secundarios. Tanto las hojas como las ramas pueden ser raíces; los nodos raíz aparecen en el nivel superior del árbol y no tienen un nodo principal. Los iconos son personalizables; para más información, consulte "Utilización de aspectos con el componente Tree" en la página 1318.

XML se puede estructurar de varias maneras, pero el componente Tree no puede utilizar todos los tipos de estructura XML. No deben anidarse atributos de nodo en un nodo secundario; cada nodo debe contener todos sus atributos necesarios. Asimismo, los atributos de cada nodo deben ser coherentes para resultar de utilidad. Por ejemplo, para describir una estructura de buzón de correo con un componente Tree, deben utilizarse los mismos atributos en cada nodo (mensaje, datos, hora, archivos adjuntos, etc.). Esto permite al árbol conocer lo que va a representar y al usuario desplazarse desde el último elemento de la jerarquía al primero y viceversa para comparar datos.

Cuando un componente Tree muestra un nodo, utiliza de forma predeterminada el atributo `label` del nodo como etiqueta de texto. Si existen otros atributos, se convierten en propiedades adicionales de los atributos del nodo dentro del árbol.

El nodo raíz real se considera como el propio componente Tree. Esto significa que el primer elemento secundario (en el ejemplo, `<node label="Mail">`), se representa como el nodo raíz en la vista de árbol. Esto demuestra que un árbol puede tener varias carpetas raíz. En este ejemplo, el árbol sólo muestra una carpeta raíz: Mail. Sin embargo, si añadiera nodos del mismo nivel a ese nivel en el XML, se visualizarían varios nodos raíz en el árbol.

Un proveedor de datos de un árbol tiene siempre preferencia por los nodos que contienen elementos secundarios que pueda mostrar. Muestra el primer elemento secundario del objeto XMLNode. Cuando se ajusta el XML en un objeto XML, la estructura tiene el siguiente aspecto:

```
<XMLDocumentObject>
  <node>
    <node label="Mail">
      <node label="INBOX"/>
      <node label="Personal Folder">
        <node label="Business" isBranch="true" />
        <node label="Demo" isBranch="true" />
        <node label="Personal" isBranch="true" />
        <node label="Saved Mail" isBranch="true" />
        <node label="bar" isBranch="true" />
      </node>
      <node label="Sent" isBranch="true" />
      <node label="Trash"/>
    </node>
  </node>
</XMLDocumentObject>
```

Flash Player ajusta los nodos XML en un nodo de documento adicional que se pasa al árbol. Cuando el árbol intenta mostrar el XML, trata de mostrar `<node>`, que no tiene ninguna etiqueta, por lo que no se muestra correctamente.

Para evitar este problema, el proveedor de datos del componente Tree debería señalar al primer elemento secundario de XMLDocumentObject, como se muestra a continuación:

```
myTree.dataProvider = myXML.firstChild;
```

Parámetros de Tree

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Tree en el inspector de propiedades o en el inspector de componentes:

multipleSelection es un valor booleano que indica si se permite la selección de varios elementos (`true`) o no se permite (`false`). El valor predeterminado es `false`.

rowHeight indica la altura de cada una de las filas, en píxeles. El valor predeterminado es 20.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente Tree utilizando sus propiedades, métodos y eventos. Para más información, consulte “Clase Tree (sólo en Flash Professional)” en la página 1318.

No se pueden especificar parámetros de datos en el inspector de propiedades ni en el inspector de componentes para el componente Tree, como se puede hacer con otros componentes. Para más información, consulte “Utilización del componente Tree (sólo en Flash Professional)” en la página 1304 y “Creación de aplicaciones con el componente Tree” en la página 1307.

Creación de aplicaciones con el componente Tree

En los siguientes procedimientos se muestra cómo utilizar el componente Tree para mostrar buzones de correo en una aplicación de correo electrónico.

El componente Tree no permite introducir parámetros de datos en el inspector de propiedades o en el inspector de componentes. La complejidad de la estructura de datos de un componente Tree obliga a importar un objeto XML en tiempo de ejecución o a crear uno mientras se edita en Flash. Para crear XML en Flash, puede utilizar la interfaz TreeDataProvider o el objeto XML de ActionScript, o crear una cadena XML. Cada una de estas opciones se describe en los siguientes procedimientos.

Para añadir un componente Tree a una aplicación y cargar XML:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. Guarde el documento como **treeMenu.fla**.
3. En el panel Componentes, haga doble clic en el componente Tree para añadirlo al escenario.
4. Seleccione la instancia de Tree. En el inspector de propiedades, introduzca el nombre de instancia **menuTree**.
5. Seleccione la instancia de Tree y presione F8. Seleccione Clip de película e introduzca el nombre **TreeNavMenu**.
6. Haga clic en el botón Avanzado y seleccione Exportar para ActionScript.
7. Escriba **TreeNavMenu** en el cuadro de texto Clase de AS 2.0 y haga clic en Aceptar.
8. Seleccione Archivo > Nuevo y, a continuación, Archivo ActionScript.
9. Guarde el archivo como **TreeNavMenu.as** en el mismo directorio donde se encuentra treeMenu.fla.

10. En la ventana Script, introduzca el siguiente código:

```
import mx.controls.Tree;

class TreeNavMenu extends MovieClip {
  var menuXML:XML;
  var menuTree:Tree;
  function TreeNavMenu() {
    // Configurar el aspecto del árbol y de los controladores de
  eventos.
    menuTree.setStyle("fontFamily", "_sans");
    menuTree.setStyle("fontSize", 12);
    // Cargar el menú XML.
    var treeNavMenu = this;
    menuXML = new XML();
    menuXML.ignoreWhite = true;
    menuXML.load("TreeNavMenu.xml");
    menuXML.onLoad = function() {
      treeNavMenu.onMenuLoaded();
    };
  }
  function change(event:Object) {
    if (menuTree == event.target) {
      var node = menuTree.selectedItem;
      // Si es una rama, expandirla o contraerla.
      if (menuTree.getIsBranch(node)) {
        menuTree.setIsOpen(node, !menuTree.getIsOpen(node), true);
      }
      // Si es un hipervínculo, saltar a él.
      var url = node.attributes.url;
      if (url) {
        getURL(url, "_top");
      }
      // Borrar cualquier selección.
      menuTree.selectedNode = null;
    }
  }
  function onMenuLoaded() {
    menuTree.dataProvider = menuXML.firstChild;
    menuTree.addEventListener("change", this);
  }
}
```

Este código ActionScript configura los estilos del árbol. Se crea un objeto XML para cargar el archivo XML que crea los nodos del árbol. A continuación, se define el controlador de eventos `onLoad` para establecer como proveedor de datos el contenido del archivo XML.

11. Cree un nuevo archivo denominado TreeNavMenu.xml en un editor de texto.
12. Introduzca el código siguiente en el archivo:

```
<node>
  <node label="My Bookmarks">
    <node label="Macromedia Web site" url="http://www.macromedia.com" />
    <node label="MXNA blog aggregator" url="http://www.markme.com/mxna"
  />
  </node>
  <node label="Google" url="http://www.google.com" />
</node>
```

13. Guarde los documentos y vuelva a treeMenu.fla. Seleccione Control > Probar película para probar la aplicación.

Para cargar el XML de un archivo externo:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. Arrastre una instancia del componente Tree al escenario.
3. Seleccione la instancia de Tree. En el inspector de propiedades, introduzca el nombre de instancia **myTree**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
var myTreeDP:XML = new XML();
myTreeDP.ignoreWhite = true;
myTreeDP.load("treeXML.xml");
myTreeDP.onLoad = function() {
  myTree.dataProvider = this.firstChild;
};
var treeListener:Object = new Object();
treeListener.change = function(evt:Object) {
  trace("selected node is: "+evt.target.selectedNode);
  trace("");
};
myTree.addEventListener("change", treeListener);
```

   Este código crea un objeto XML denominado `myTreeDP` y llama al método `XML.load()` para cargar un origen de datos XML. A continuación, el código define un controlador de eventos `onLoad` que establece la propiedad `dataProvider` de la instancia `myTree` en el nuevo objeto XML cuando se carga XML. Para más información sobre el objeto XML, consulte su entrada en *Referencia del lenguaje ActionScript 2.0*.
5. Cree un nuevo archivo denominado **treeXML.xml** en un editor de texto.

6. Introduzca el código siguiente en el archivo:

```
<node>
  <node label="Mail">
    <node label="INBOX"/>
    <node label="Personal Folder">
      <node label="Business" isBranch="true" />
      <node label="Demo" isBranch="true" />
      <node label="Personal" isBranch="true" />
      <node label="Saved Mail" isBranch="true" />
      <node label="bar" isBranch="true" />
    </node>
    <node label="Sent" isBranch="true" />
    <node label="Trash"/>
  </node>
</node>
```

7. Seleccione Control > Probar película.

   En el archivo SWF, puede visualizar la estructura de XML que se muestra en el árbol. Haga clic en los elementos del árbol para ver cómo las sentencias `trace()` del controlador de eventos `change` envían los valores de datos al panel Salida.

Para utilizar la clase TreeDataProvider para crear XML en Flash durante la edición:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. Arrastre una instancia del componente Tree al escenario.
3. Seleccione la instancia de Tree e introduzca en el inspector de propiedades el nombre de instancia **myTree**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
var myTreeDP:XML = new XML();
myTreeDP.addTreeNode("Local Folders", 0);
// Utilizar XML.firstChild para anidar nodos secundarios en Local
  Folders.
var myTreeNode:XMLNode = myTreeDP.firstChild;
myTreeNode.addTreeNode("Inbox", 1);
myTreeNode.addTreeNode("Outbox", 2);
myTreeNode.addTreeNode("Sent Items", 3);
myTreeNode.addTreeNode("Deleted Items", 4);
// Asignar el origen de datos myTreeDP al componente myTree.
myTree.dataProvider = myTreeDP;
// Configurar cada uno de los cuatro nodos secundarios para que sean
  ramas.
for (var i = 0; i<myTreeNode.childNodes.length; i++) {
  var node:XMLNode = myTreeNode.getTreeNodeAt(i);
  myTree.setIsBranch(node, true);
}
```

Este código crea un objeto XML denominado `myTreeDP`. Cualquier objeto XML que se encuentre en el mismo fotograma que un componente Tree recibe automáticamente todas las propiedades y los métodos de la interfaz TreeDataProvider. La segunda línea de código crea un único nodo raíz denominado Local Folders. Para obtener información detallada sobre el resto del código, vea los comentarios (líneas precedidas por `//`) que se encuentran en todo el código.

5. Seleccione Control > Probar película.

   En el archivo SWF, puede visualizar la estructura de XML que se muestra en el árbol. Haga clic en los elementos del árbol para ver cómo las sentencias `trace()` del controlador de eventos `change` envían los valores de datos al panel Salida.

Para utilizar la clase XML de ActionScript para crear XML:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. Arrastre una instancia del componente Tree al escenario.
3. Seleccione la instancia de Tree. En el inspector de propiedades, introduzca el nombre de instancia **myTree**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
// Crear un objeto XML.
var myTreeDP:XML = new XML();
// Crear los valores de nodo.
var myNode0:XMLNode = myTreeDP.createElement("node");
myNode0.attributes.label = "Local Folders";
myNode0.attributes.data = 0;
var myNode1:XMLNode = myTreeDP.createElement("node");
myNode1.attributes.label = "Inbox";
myNode1.attributes.data = 1;
var myNode2:XMLNode = myTreeDP.createElement("node");
myNode2.attributes.label = "Outbox";
myNode2.attributes.data = 2;
var myNode3:XMLNode = myTreeDP.createElement("node");
myNode3.attributes.label = "Sent Items";
myNode3.attributes.data = 3;
var myNode4:XMLNode = myTreeDP.createElement("node");
myNode4.attributes.label = "Deleted Items";
myNode4.attributes.data = 4;
// Asignar nodos a la jerarquía del árbol XML.
myTreeDP.appendChild(myNode0);
myTreeDP.firstChild.appendChild(myNode1);
myTreeDP.firstChild.appendChild(myNode2);
myTreeDP.firstChild.appendChild(myNode3);
myTreeDP.firstChild.appendChild(myNode4);
// Asignar el origen de datos myTreeDP al componente Tree.
myTree.dataProvider = myTreeDP;
```

   Para más información sobre el objeto XML, consulte su entrada en *Referencia del lenguaje ActionScript 2.0*.

5. Seleccione Control > Probar película.

   En el archivo SWF, puede visualizar la estructura de XML que se muestra en el árbol. Haga clic en los elementos del árbol para ver cómo las sentencias `trace()` del controlador de eventos `change` envían los valores de datos al panel Salida.

Para utilizar una cadena bien formada que permita crear XML en Flash durante la edición:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
2. Arrastre una instancia del componente Tree al escenario.
3. Seleccione la instancia de Tree. En el inspector de propiedades, introduzca el nombre de instancia **myTree**.
4. En el panel Acciones del fotograma 1, introduzca el código siguiente:

```
var myTreeDP:XML = new XML("<node label='Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
myTree.dataProvider = myTreeDP;
```

   Este código crea el objeto XML `myTreeDP` y lo asigna a la propiedad `dataProvider` de `myTree`.

5. Seleccione Control > Probar película.

   En el archivo SWF, puede visualizar la estructura de XML que se muestra en el árbol. Haga clic en los elementos del árbol para ver cómo las sentencias `trace()` del controlador de eventos `change` envían los valores de datos al panel Salida.

Personalización del componente Tree (sólo en Flash Professional)

El componente Tree puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`). Cuando un árbol no es suficientemente ancho para mostrar el texto de los nodos, el texto se recorta.

Utilización de estilos con el componente Tree

El componente Tree utiliza los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Ambos	Color de fondo de la lista. El color predeterminado es blanco y se define en la declaración de estilo de clase. Este estilo se omite si se especifica `alternatingRowColors`.
`backgroundDisabledColor`	Ambos	Color de fondo cuando el valor de la propiedad `enabled` del componente es `"false"`. El valor predeterminado es 0xDDDDDD (gris medio).
`depthColors`	Ambos	Establece los colores de fondo de las filas basándose en la profundidad de cada nodo. El valor es una matriz de colores cuyo primer elemento es el color de fondo del nodo raíz, cuyo segundo elemento es el color de fondo de sus nodos secundarios, y así sucesivamente, hasta completar el número de colores proporcionados por la matriz. Esta propiedad de estilo no se establece de forma predeterminada.
`borderStyle`	Ambos	El componente Tree utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El estilo de borde predeterminado es `"inset"`.
`color`	Ambos	Color del texto.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.

Estilo	Tema	Descripción
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.
`defaultLeafIcon`	Ambos	Icono que se muestra en un control de árbol para los nodos hoja cuando no se especifica ningún icono para un nodo determinado. El valor predeterminado es `"TreeNodeIcon"`, que es una imagen que representa un pedazo de papel.
`disclosureClosedIcon`	Ambos	Icono que se muestra junto a un nodo de carpeta cerrada en un componente Tree. El valor predeterminado es `"TreeDisclosureClosed"`, que es una flecha gris que señala hacia la derecha.
`disclosureOpenIcon`	Ambos	Icono que se muestra junto a un nodo de carpeta abierta en un componente Tree. El valor predeterminado es `"TreeDisclosureOpen"`, que es una flecha gris que señala hacia abajo.
`folderClosedIcon`	Ambos	Icono que se muestra para un nodo de carpeta cerrada en un componente Tree si no se establece ningún icono específico del nodo. El valor predeterminado es `"TreeFolderClosed"`, que es una imagen de carpeta de archivos cerrada, de color amarillo.

Estilo	Tema	Descripción
`folderOpenIcon`	Ambos	Icono que se muestra para un nodo de carpeta abierta en un componente Tree si no se establece ningún icono específico del nodo. El valor predeterminado es `"TreeFolderOpen"`, que es una imagen de carpeta de archivos abierta, de color amarillo.
`indentation`	Ambos	Número de píxeles de la sangría de cada fila de un componente Tree. El valor predeterminado es 17.
`openDuration`	Ambos	Duración, en milisegundos, de las animaciones para expandir y contraer. El valor predeterminado es 250.
`openEasing`	Ambos	Referencia a una función de interpolación que controla las animaciones para expandir y contraer. De forma predeterminada es una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.
`repeatDelay`	Ambos	Número de milisegundos de demora entre el momento en que el usuario presiona por primera vez un botón en la barra de desplazamiento y el momento en que comienza a repetirse la acción. El valor predeterminado es 500 (medio segundo).
`repeatInterval`	Ambos	Número de milisegundos entre clics automáticos cuando un usuario mantiene presionado el botón del ratón sobre un botón de la barra de desplazamiento. El valor predeterminado es 35.
`rollOverColor`	Ambos	Color de fondo de una fila por la que se desplaza el puntero. El valor predeterminado es 0xE3FFD6 (verde brillante) con el tema Halo y 0xAAAAAA (gris claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a setStyle(), el marco establece `rollOverColor` en un valor relacionado con el valor de `themeColor` elegido.
`selectionColor`	Ambos	Color de fondo de una fila seleccionada. El valor predeterminado es 0xCDFFC1 (verde claro) con el tema Halo y 0xEEEEEE (gris muy claro) con el tema Sample. Cuando se cambia `themeColor` en una llamada a setStyle(), el marco establece `selectionColor` en un valor relacionado con el valor de `themeColor` elegido.

Estilo	Tema	Descripción
`selectionDuration`	Ambos	Longitud de la transición desde un estado normal a un estado seleccionado o viceversa, en milisegundos. El valor predeterminado es 200.
`selectionDisabledColor`	Ambos	Color de fondo de una fila seleccionada. El valor predeterminado es 0xDDDDDD (gris medio). Como el valor predeterminado de esta propiedad coincide con el valor predeterminado de `backgroundDisabledColor`, la selección no es visible cuando se desactiva el componente, a menos que se cambie una de estas propiedades de estilo.
`selectionEasing`	Ambos	Referencia a la ecuación de suavizado que se utiliza para controlar la transición entre los estados de selección. Se aplica únicamente a la transición de un estado normal a un estado seleccionado. La ecuación predeterminada usa una función sinusoidal entrante/saliente. Para más información, consulte "Personalización de animaciones de componentes" en *Utilización de componentes*.
`textRollOverColor`	Ambos	Color del texto cuando el puntero se desplaza sobre él. El valor predeterminado es 0x2B333C (gris oscuro). Este estilo es importante cuando se establece `rollOverColor` porque ambos colores deben complementarse para que el texto pueda verse con claridad al desplazar el puntero sobre él.
`textSelectedColor`	Ambos	Color del texto de la fila seleccionada. El valor predeterminado es 0x005F33 (gris oscuro). Este estilo es importante cuando se establece `selectionColor` porque ambos colores deben complementarse para que el texto pueda verse con claridad cuando se selecciona.
`useRollOver`	Ambos	Determina si el resaltado se activa cuando el puntero se desplaza sobre una fila. El valor predeterminado es `true`.

Por ejemplo, el siguiente código crear una instancia `first_tr` de Tree mediante `UIObject.createClassObject()`, llena el árbol mediante un proveedor de datos y utiliza `UIObject.setStyle()` para cambiar el sangrado de los nodos del árbol por 8 píxeles. Arrastre un componente Tree a la biblioteca del documento actual y añada lo siguiente al primer fotograma de la línea de tiempo principal:

```
import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 20);
first_tr.setSize(200, 100);
first_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folder'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/><node label='2nd
  Local Folder'><node label='Inbox' data='0'/><node label='Outbox'
  data='1'/></node></node>");
first_tr.dataProvider = trDP_xml;

first_tr.setStyle("indentation", 8);
```

Definición de estilos de todos los componentes Tree de un documento

La clase Tree hereda de la clase List, que a su vez hereda de la clase ScrollSelectList. Las propiedades de estilo de clase predeterminadas se definen en la clase ScrollSelectList, que amplían el componente Menu y todos los componentes basados en List. Puede definir directamente en esta clase nuevos valores de estilo predeterminados y los nuevos valores se reflejan en todos los componentes afectados.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Para definir una propiedad de estilo sólo en el componente Tree, puede crear una nueva instancia `CSSStyleDeclaration` y almacenarla en `_global.styles.DataGrid`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.Tree == undefined) {
  _global.styles.Tree = new CSSStyleDeclaration();
}
_global.styles.Tree.setStyle("backgroundColor", 0xFF00AA);
```

Cuando se crea una nueva declaración de estilo de clase, se pierden todos los valores predeterminados proporcionados por la declaración `ScrollSelectList`. Entre ellos, el valor de `backgroundColor`, necesario para la compatibilidad con eventos de ratón. Para crear una declaración de estilo de clase y conservar los valores predeterminados, utilice un bucle `for` para copiar los valores anteriores a la nueva declaración.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.Tree;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Utilización de aspectos con el componente Tree

El componente Tree utiliza una instancia de RectBorder para su borde y barras de desplazamiento para desplazarse por las imágenes. Para más información sobre la aplicación de aspectos en estos elementos visuales, consulte "Clase RectBorder" en la página 1103 y "Utilización de aspectos con el componente UIScrollBar" en la página 1435.

Clase Tree (sólo en Flash Professional)

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > ScrollSelectList > Componente List > Tree

Nombre de clase de ActionScript mx.controls.Tree

Los métodos, propiedades y eventos de la clase Tree permiten administrar y manipular objetos Tree.

Resumen de métodos de la clase Tree

En la tabla siguiente se enumeran los métodos de la clase Tree.

Método	Descripción
`Tree.addTreeNode()`	Añade un nodo a una instancia de Tree.
`Tree.addTreeNodeAt()`	Añade un nodo en una ubicación específica de una instancia de Tree.
`Tree.getDisplayIndex()`	Devuelve el índice de visualización de un nodo determinado.
`Tree.getIsBranch()`	Especifica si la carpeta es una rama (cuenta con un icono de carpeta y una flecha de ampliación).
`Tree.getIsOpen()`	Indica si un nodo está abierto o cerrado.
`Tree.getNodeDisplayedAt()`	Asigna un índice de visualización del árbol al nodo que se muestra en esa posición.
`Tree.getTreeNodeAt()`	Devuelve un nodo en la raíz del árbol.
`Tree.refresh()`	Actualiza el árbol.
`Tree.removeAll()`	Elimina todos los nodos de la instancia de Tree y actualiza el árbol.
`Tree.removeTreeNodeAt()`	Elimina un nodo en una posición específica y actualiza el árbol.
`Tree.setIcon()`	Especifica un icono para el nodo especificado.

Método	Descripción
`Tree.setIsBranch()`	Especifica si un nodo es una rama (cuenta con un icono de carpeta y una flecha de ampliación).
`Tree.setIsOpen()`	Abre o cierra un nodo.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Tree de la clase UIObject. Al llamar a estos métodos desde el objeto Tree, debe utilizarse la forma *`TreeInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Tree de la clase UIComponent. Al llamar a estos métodos desde el objeto Tree, debe utilizarse la forma *TreeInstance.methodName*.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Métodos heredados de la clase List

En la tabla siguiente se enumeran los métodos que hereda la clase Tree de la clase List. Al llamar a estos métodos desde el objeto Tree, debe utilizarse la forma *TreeInstance.methodName*.

Método	Descripción
`List.addItem()`	Añade un elemento al final de la lista.
`List.addItemAt()`	Añade un elemento a la lista en el índice especificado. Con el componente Tree, es preferible utilizar `Tree.addTreeNodeAt()`.
`List.getItemAt()`	Devuelve el elemento del índice especificado.
`List.removeAll()`	Elimina todos los elementos de la lista.
`List.removeItemAt()`	Elimina el elemento en el índice especificado.
`List.replaceItemAt()`	Sustituye por otro el elemento del índice especificado.
`List.setPropertiesAt()`	Aplica las propiedades especificadas al elemento especificado.
`List.sortItems()`	Ordena los elementos de la lista según la función de comparación especificada.
`List.sortItemsBy()`	Ordena los elementos de la lista según la propiedad especificada.

Resumen de propiedades de la clase Tree

En la tabla siguiente se enumeran las propiedades de la clase Tree.

Propiedad	Descripción
`Tree.dataProvider`	Especifica un origen de datos XML.
`Tree.firstVisibleNode`	Especifica el primer nodo en la parte superior de la pantalla.
`Tree.selectedNode`	Especifica el nodo seleccionado en una instancia de Tree.
`Tree.selectedNodes`	Especifica los nodos seleccionados en una instancia de Tree.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Tree de la clase UIObject. Al acceder a estas propiedades desde el objeto Tree, debe utilizarse la forma *`TreeInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Sólo lectura; anchura del objeto, expresada en píxeles.
`UIObject.x`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.y`	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Tree de la clase UIComponent. Al acceder a estas propiedades desde el objeto Tree, debe utilizarse la forma *TreeInstance.propertyName*.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase List

En la tabla siguiente se enumeran las propiedades que hereda la clase Tree de la clase List. Al acceder a estas propiedades desde el objeto Tree, debe utilizarse la forma *TreeInstance.propertyName*.

Propiedad	Descripción
`List.cellRenderer`	Asigna la clase o el símbolo que se va utilizar para mostrar cada fila de la lista.
`List.dataProvider`	Origen de los elementos de la lista.
`List.hPosition`	Posición horizontal de la lista.
`List.hScrollPolicy`	Indica si la barra de desplazamiento horizontal se muestra (`"on"`) o no (`"off"`).
`List.iconField`	Campo de cada elemento que se utiliza para especificar iconos.
`List.iconFunction`	Función que determina qué icono se debe utilizar.
`List.labelField`	Especifica el campo de cada elemento que se utilizará como texto de etiqueta.
`List.labelFunction`	Función que determina qué campos de cada elemento se utilizarán para el texto de etiqueta.
`List.length`	Número de elementos de la lista. Es una propiedad de sólo lectura.
`List.maxHPosition`	Número de píxeles que la lista puede desplazarse a la derecha cuando `List.hScrollPolicy` está definida en `"on"`.
`List.multipleSelection`	Indica si en la lista se permite la selección múltiple (`true`) o no (`false`).

Propiedad	Descripción
List.rowCount	Número de filas que son al menos parcialmente visibles en la lista.
List.rowHeight	Altura de las filas de la lista, expresada en píxeles.
List.selectable	Indica si la lista es seleccionable (true) o no (false).
List.selectedIndex	Índice de una selección en una lista de selección única.
List.selectedIndices	Matriz de los elementos seleccionados en una lista de selección múltiple.
List.selectedItem	Elemento seleccionado en una lista de selección única. Es una propiedad de sólo lectura.
List.selectedItems	Objetos del elemento seleccionado en una lista de selección múltiple. Es una propiedad de sólo lectura.
List.vPosition	Desplaza la lista para que el elemento visible en la parte superior sea el número asignado.
List.vScrollPolicy	Indica si la barra de desplazamiento vertical se muestra ("on"), no se muestra ("off") o se muestra cuando es necesario ("auto").

Resumen de eventos de la clase Tree

En la tabla siguiente se enumeran los eventos de la clase Tree.

Evento	Descripción
Tree.nodeClose	Se difunde cuando un usuario cierra un nodo.
Tree.nodeOpen	Se difunde cuando un usuario abre un nodo.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Tree de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.

Evento	Descripción
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Tree de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase List

En la tabla siguiente se enumeran los eventos que hereda la clase Tree de la clase List.

Evento	Descripción
List.change	Se difunde cuando la interacción del usuario provoca un cambio en la selección.
List.itemRollOut	Se difunde cuando el puntero se desplaza sobre elementos de la lista y después sale de ellos.
List.itemRollOver	Se difunde cuando el puntero se desplaza sobre un elemento.
List.scroll	Se difunde cuando se recorre la lista.

Tree.addTreeNode()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

Sintaxis 1:

```
treeInstance.addTreeNode(label [, data])
```

Sintaxis 2:

```
treeInstance.addTreeNode(child)
```

Parámetros

label Cadena que muestra el nodo o un objeto con un campo de etiqueta (o con cualquier nombre de campo de etiqueta que especifique la propiedad `labelField`).

data Objeto de cualquier tipo que está asociado con el nodo. Este parámetro es opcional.

child Cualquier objeto XMLNode.

Valor devuelto

El nodo XML añadido.

Descripción

Método; añade un nodo secundario al árbol. El nodo se crea a partir de la información proporcionada en los parámetros *label* y *data* (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto XMLNode (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

Al llamar a este método se actualiza la vista.

Ejemplo

En el siguiente ejemplo se crean dos componentes Tree con un nodo para cada uno, `1st Local Folders` y `2nd Local Folders`, respectivamente. A continuación, se añade el nodo (`2nd Local Folders`) del segundo componente Tree al primero y, además, se añade un nuevo nodo denominado `Inbox`.

Primero debe añadir el componente a la biblioteca de documentos; para ello, arrastre un componente Tree al escenario y elimínelo a continuación. Luego añada el siguiente código al fotograma 1.

SUGERENCIA	Realice primero el ejemplo sin las dos sentencias `addTreeNode()` finales; a continuación, intente realizar el ejemplo completo.

```
/**
 Se requiere:
  - Componente Tree en biblioteca
*/

import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 10);
first_tr.setSize(200, 100);

this.createClassObject(Tree, "second_tr", 20);
second_tr.setSize(200, 100);
second_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
first_tr.dataProvider = trDP_xml;

var trDP2_xml:XML = new XML("<node label='2nd Local Folders'><node
  label='Outbox' data='0'/><node label='Outbox' data='1'/></node>");
second_tr.dataProvider = trDP2_xml;

// Añadir el nodo de second_tr a first_tr.
first_tr.addTreeNode(second_tr.getTreeNodeAt(0));

// Añadir nodo a first_tr.
first_tr.addTreeNode("Inbox", "data");
```

Tree.addTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

Sintaxis 1:

```
treeInstance.addTreeNodeAt(index, label [, data])
```

Sintaxis 2:

```
treeInstance.addTreeNodeAt(index, child)
```

Parámetros

index Posición de índice basado en cero (entre los nodos secundarios) en la que debe añadirse el nodo.

label Cadena que contiene el nombre del nodo que se va a añadir.

data Objeto de cualquier tipo que está asociado con el nodo. Este parámetro es opcional.

child Cualquier objeto XMLNode.

Valor devuelto

El nodo XML añadido.

Descripción

Método; añade un nodo a la ubicación especificada del árbol. El nodo se crea a partir de la información proporcionada en los parámetros *label* y *data* (Sintaxis 1) o a partir del objeto XMLNode creado previamente (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

Al llamar a este método se actualiza la vista.

Ejemplo

En el siguiente ejemplo se crean dos componentes Tree con un nodo cada uno: `1st Local Folders` y `2nd Local Folders`, respectivamente. Se utiliza la primera sintaxis de `addTreeNodeAt()` para añadir un nuevo nodo, `Inbox`, al primero componente Tree. A continuación, se utiliza la segunda sintaxis de `addTreeNodeAt()` para añadir el primer nodo (`(getTreeNodeAt(0)`) del segundo componente Tree al primero.

Primero debe añadir el componente a la biblioteca de documentos; para ello, arrastre un componente Tree al escenario y elimínelo a continuación. Luego añada el siguiente código al fotograma 1.

SUGERENCIA	Realice primero el ejemplo sin las dos sentencias `addTreeNodeAt()` finales; a continuación, intente realizar el ejemplo completo.

```
/**
 Se requiere:
  - Componente Tree en biblioteca
*/

import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 10);
first_tr.setSize(200, 100);

this.createClassObject(Tree, "second_tr", 20);
second_tr.setSize(200, 100);
second_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
first_tr.dataProvider = trDP_xml;

var trDP2_xml:XML = new XML("<node label='2nd Local Folders'><node
  label='Outbox' data='0'/><node label='Outbox' data='1'/></node>");
second_tr.dataProvider = trDP2_xml;

// Añadir nodo a first_tr.
first_tr.addTreeNodeAt(1, "Inbox", "data");
// Añadir el nodo de second_tr a first_tr.
first_tr.addTreeNodeAt(2, second_tr.getTreeNodeAt(0));
```

Tree.dataProvider

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`treeInstance.dataProvider`

Descripción

Propiedad; XML o una cadena. Si `dataProvider` es un objeto XML, se añade directamente al árbol. Si `dataProvider` es una cadena, debe contener XML válido que el árbol pueda leer y convertir en un objeto XML.

Puede cargar XML desde un origen externo en tiempo de ejecución o crearlo en Flash durante la edición. Para crear XML, puede utilizar los métodos de TreeDataProvider o los métodos y las propiedades de la clase XML de ActionScript incorporados. También puede crear una cadena que contenga XML.

Los objetos XML que se encuentran en el mismo fotograma que un componente Tree contienen automáticamente los métodos y las propiedades de TreeDataProvider. Puede utilizar el objeto XML o XMLNode de ActionScript.

Ejemplo

En el siguiente ejemplo se utiliza la propiedad `dataProvider` para añadir el contenido de un archivo XML a la instancia `my_tr` del componente Tree:

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function(success:Boolean){
   my_tr.dataProvider = trDP_xml.firstChild;
}
trDP_xml.load("http://www.flash-mx.com/mm/xml/tree.xml");
```

NOTA La mayoría de archivos XML contienen espacios en blanco. Para que Flash omita los espacios en blanco, debe definirse la propiedad `XML.ignoreWhite` en `true`.

Tree.firstVisibleNode

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.firstVisibleNode = someNode
```

Descripción

Propiedad; indica el primer nodo visible en la parte superior de la visualización del árbol. Utilice esta propiedad para desplazarse hasta la posición que se desee en la visualización del árbol. Si el nodo especificado *someNode* se encuentra en un nodo que no se ha ampliado, la definición de `firstVisibleNode` no tiene ningún efecto. El valor predeterminado es el primer nodo visible o `undefined`, en el caso de que no haya ningún nodo visible. El valor de esta propiedad es un objeto XMLNode.

Esta propiedad es análoga a la propiedad `List.vPosition`.

Ejemplo

En el siguiente ejemplo se llena un componente Tree denominado `my_tr` con seis nodos que crea al componente a partir de una cadena de texto XML. Para que el último nodo sea visible en el componente Tree, se asigna el último nodo (posición relativa 5) a la propiedad `firstVisibleNode`. Para hacer que otros nodos sean visibles, cambie el 5 por otros valores de 0 a 4.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='3rd Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='4th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='5th Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='6th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node>");
my_tr.dataProvider = trDP_xml;

// Definir último nodo como nodo visible.
my_tr.firstVisibleNode = my_tr.getTreeNodeAt(5);
```

Tree.getDisplayIndex()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.getDisplayIndex(node)
```

Parámetros

node Objeto XMLNode.

Valor devuelto

El índice del nodo especificado o `undefined` si el nodo no se está mostrando en ese momento.

Descripción

Método; devuelve el índice de visualización del nodo especificado en el parámetro *node*.

El índice de visualización indica la posición del elemento en la lista de elementos visibles en la ventana del árbol. Por ejemplo, ningún nodo secundario de un nodo cerrado se incluye en el índice de visualización. La lista de índices de visualización empieza por 0 y sigue por los elementos visibles independientemente de su nodo principal. En otras palabras, el índice es el número de fila, empezando por 0, de las filas visualizadas.

Ejemplo

En el siguiente ejemplo se añaden seis nodos a un componente Tree y se llama a `getDisplayIndex()` para visualizar la posición del nodo que selecciona el usuario.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 140);
```

```
var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='3rd Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='4th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='5th Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='6th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node>");
my_tr.dataProvider = trDP_xml;
my_tr.firstVisibleNode = my_tr.getTreeNodeAt(0);

my_tr.addEventListener("change", listChanged);
function listChanged(evt_obj:Object) {
  trace(my_tr.getDisplayIndex(evt_obj.target.selectedNode));
}
```

Tree.getIsBranch()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.getIsBranch(node)
```

Parámetros

node Objeto XMLNode.

Valor devuelto

Valor booleano que indica si el nodo es una rama (`true`) o no (`false`).

Descripción

Método; indica si el nodo especificado cuenta con un icono de carpeta y una flecha de ampliación (por tanto, es una rama). Esto se establece de forma automática al añadir nodos secundarios al nodo. Sólo debe llamar a `Tree.setIsBranch()` para crear carpetas vacías.

Ejemplo

En el siguiente ejemplo se crean dos nodos en un componente Tree y se llama a `isBranch()` para determinar si el segundo nodo es una rama.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

var isBranch:Boolean = my_tr.getIsBranch(my_tr.getTreeNodeAt(1));
trace("2nd node is a branch: " + isBranch);
```

Véase también

`Tree.setIsBranch()`

Tree.getIsOpen()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`treeInstance.getIsOpen(`*`node`*`)`

Parámetros

`node` Objeto XMLNode.

Valor devuelto

Valor booleano que indica si el árbol está abierto (`true`) o cerrado (`false`).

Descripción

Método; indica si el nodo especificado está abierto o cerrado.

NOTA	Los índices de visualización cambian cada vez que los nodos se abren y se cierran.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a un componente Tree denominado `my_tr`, se abre el segundo nodo y, a continuación, se llama a `getIsOpen()` para visualizar el estado del segundo nodo.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;
my_tr.setIsOpen(my_tr.getTreeNodeAt(1), true);
var isOpen:Boolean = my_tr.getIsOpen(my_tr.getTreeNodeAt(1));
trace("2nd node is a open: " + isOpen);
```

Tree.getNodeDisplayedAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`treeInstance.getNodeDisplayedAt(`*`index`*`)`

Parámetros

index Entero que representa la posición de visualización en el área visualizable del árbol. La serie de números está basada en cero; el nodo de la primera posición es 0, la segunda posición es 1, etc.

Valor devuelto

Objeto XMLNode especificado.

Descripción

Método; asigna un índice de visualización del árbol al nodo que se muestra en esa posición. Por ejemplo, si la quinta fila del árbol muestra un nodo que se encuentra en el octavo nivel de la jerarquía, dicho nodo se devolverá mediante una llamada a `getNodeDisplayedAt(4)`.

El índice de visualización es una matriz de elementos que se pueden visualizar en la ventana del árbol. Por ejemplo, ningún nodo secundario de un nodo cerrado se incluye en el índice de visualización. El índice de visualización empieza por 0 y sigue por los elementos visibles independientemente de su nodo principal. En otras palabras, el índice de visualización es el número de fila, empezando por 0, de las filas visualizadas.

NOTA	Los índices de visualización cambian cada vez que los nodos se abren y se cierran.

Ejemplo

En el siguiente ejemplo se añade un nodo a una instancia de Tree denominada `my_tr` y, a continuación, se llama al método `getNodeDisplayedAt()` para recuperar el nodo en la posición de visualización 0 (cero). En el ejemplo se llama a la función trace() para visualizar el nodo.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label=\"1st Local Folders\"><node
  label=\"Inbox\" data=\"0\" />");
my_tr.dataProvider = trDP_xml;

trace(my_tr.getNodeDisplayedAt(0));
```

Tree.getTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`treeInstance.getTreeNodeAt(index)`

Parámetros

index Número de índice de un nodo.

Valor devuelto

Un objeto XMLNode.

Descripción

Método; devuelve el nodo especificado en la raíz de `myTree`.

Ejemplo

En el siguiente ejemplo se añade un nodo a una instancia de Tree denominada `my_tr` y, a continuación, se llama al método `getTreeNodeAt()` para devolver el primer nodo del árbol.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label=\"1st Local Folders\"><node
  label=\"Inbox\" data=\"0\" /><node label=\"Outbox\" data=\"1\" /></
  node>");
my_tr.dataProvider = trDP_xml;

trace(my_tr.getTreeNodeAt(0));
```

Tree.nodeClose

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
var listenerObject:Object = new Object();
listenerObject.nodeClose = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
treeInstance.addEventListener("nodeClose", listenerObject);
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario cierra los nodos de un componente Tree.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. El componente Tree difunde un evento `nodeClose` cuando se cierra uno de sus nodos con un clic. El evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Tree.nodeClose` tiene una propiedad adicional: `node` (el nodo XML que se ha cerrado).

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a la instancia de Tree `my_tr` y, a continuación, se crean dos objetos detectores: uno para eventos `nodeOpen` y otro para eventos `nodeClose`. Cuando se producen dichos eventos, la función detectora utiliza una sentencia trace para visualizar el evento y el nodo afectado en el panel Salida.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Crear un objeto detector.
var trListener:Object = new Object();
trListener.nodeOpen = function(evt_obj:Object){
   trace("Node opened\n" + evt_obj.node);
   trace("\n");
}
trListener.nodeClose = function(evt_obj:Object){
   trace("Node closed\n" + evt_obj.node);
   trace("\n");
}
// Añadir detectores.
my_tr.addEventListener("nodeOpen", trListener);
my_tr.addEventListener("nodeClose", trListener);
```

Tree.nodeOpen

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
var listenerObject:Object = new Object();
listenerObject.nodeOpen = function(eventObject:Object) {
  // Introducir aquí el código propio.
};
treeInstance.addEventListener("nodeOpen", listenerObject);
```

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario abre un nodo de un componente Tree.

Los componentes de la versión 2 utilizan un modelo de evento distribuidor/detector. El componente Tree distribuye un evento `nodeOpen` cuando un usuario abre un nodo con un clic. El evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. El objeto del evento `Tree.nodeOpen` tiene una propiedad adicional: `node` (el nodo XML que se ha abierto).

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a la instancia de Tree `my_tr` y, a continuación, se crean dos objetos detectores: uno para eventos `nodeOpen` y otro para eventos `nodeClose`. Cuando se producen dichos eventos, las funciones detectoras realizan llamadas a sentencias trace para visualizar el evento y el nodo afectado en el panel Salida.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;
```

```
// Crear un objeto detector.
var trListener:Object = new Object();
trListener.nodeOpen = function(evt_obj:Object){
   trace("Node opened\n" + evt_obj.node);
   trace("\n");
}
trListener.nodeClose = function(evt_obj:Object){
   trace("Node closed\n" + evt_obj.node);
   trace("\n");
}
// Añadir detectores.
my_tr.addEventListener("nodeOpen", trListener);
my_tr.addEventListener("nodeClose", trListener);
```

Tree.refresh()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.refresh()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; actualiza el árbol.

Ejemplo

En el siguiente ejemplo se añade un nodo a una instancia de Tree denominada my_tr y se crean detectores para un botón Refresh (Actualizar) y un botón Remove All (Quitar todos). Suponiendo que el origen XML del proveedor de datos ha cambiado, el usuario puede hacer clic en el botón Refresh y el código llamará al método `refresh()` para actualizar el contenido del árbol.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`. A continuación, añada un botón denominado `refresh_button`. Luego añada el código siguiente al fotograma 1 de la línea de tiempo principal:

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
  - Componente Button en el escenario (nombre de instancia: refresh_button)
*/

var my_tr:mx.controls.Tree;
var refresh_button:mx.controls.Button;
var removeAll_button:mx.controls.Button;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function() {
 my_tr.dataProvider = this.firstChild;
};
trDP_xml.load("http://yourXMLsourcehere");

function refreshListener(evt_obj:Object):Void {
 my_tr.refresh();
}
refresh_button.addEventListener("click", refreshListener);
```

Tree.removeAll()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.removeAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los nodos y actualiza el árbol.

Ejemplo

En el siguiente ejemplo se añade un nodo a una instancia de Tree denominada my_tr y se crea un detector para un botón Remove All (Quitar todos). Cuando el usuario hace clic en el botón Remove All, el código llama al método `removeAll()` para actualizar el árbol.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr` y añadir un botón denominado `removeAll_button`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
  - Componente Button en el escenario (nombre de instancia: refresh_button)
  - Componente Button en el escenario (nombre de instancia:
  removeAll_button)
*/

var my_tr:mx.controls.Tree;
var removeAll_button:mx.controls.Button;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function() {
 my_tr.dataProvider = this.firstChild;
};
trDP_xml.load("http://www.flash-mx.com/mm/xml/tree.xml");

function removeAllListener(evt_obj:Object):Void {
 my_tr.removeAll();
}
removeAll_button.addEventListener("click", removeAllListener);
```

Tree.removeTreeNodeAt()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.removeTreeNodeAt(index)
```

Parámetros

index Número de índice de un nodo secundario de un árbol. Cada nodo secundario de un árbol tiene asignado un índice basado en cero en el orden en que se ha creado.

Valor devuelto

Un objeto XMLNode o `undefined` si se produce algún error.

Descripción

Método; elimina un nodo (especificado por su posición de índice) en la raíz del árbol y lo actualiza.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a una instancia de Tree y se crea un detector para un evento change en el árbol. Cuando se produce un evento change, las funciones detectoras realizan una llamada al método `removeTreeNodeAt()` para eliminar el nodo seleccionado del árbol.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

var treeListener:Object = new Object();
treeListener.change = function (evt_obj:Object) {
 my_tr.removeTreeNodeAt(my_tr.selectedIndex);
}
my_tr.addEventListener("change", treeListener);
```

Tree.selectedNode

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.selectedNode
```

Descripción

Propiedad; especifica el nodo seleccionado en una instancia de árbol.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a una instancia de Tree y se establece el estado seleccionado en el segundo nodo.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

// Seleccionar el segundo nodo.
my_tr.selectedNode = my_tr.getTreeNodeAt(1);
```

Véase también

`Tree.selectedNodes`

Tree.selectedNodes

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.selectedNodes
```

Descripción

Propiedad; especifica los nodos seleccionados en una instancia de árbol.

Ejemplo

En el siguiente ejemplo se añaden tres nodos a una instancia de Tree y se establece el estado seleccionado en los dos primeros.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node><node label='3rd Local Folders'><node
  label='Inbox' data='2'/><node label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Permitir varias selecciones.
my_tr.multipleSelection = true;

// Seleccionar el primer nodo y el segundo.
my_tr.selectedNodes = [my_tr.getTreeNodeAt(0), my_tr.getTreeNodeAt(1)];
```

Véase también

`Tree.selectedNode`

Tree.setIcon()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

```
treeInstance.setIcon(node, linkID [, linkID2])
```

Parámetros

node Nodo XML.

linkID Identificador de vinculación de un símbolo que va a utilizarse como icono junto al nodo. Este parámetro se utiliza para los nodos hoja y para el estado cerrado de los nodos rama.

linkID2 En un nodo rama, identificador de vinculación de un símbolo que va a utilizarse como icono para representar el nodo abierto. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; especifica un icono para el nodo especificado. Este método toma un parámetro ID (*linkID*) para los nodos hoja y dos parámetros ID (*linkID* y *linkID2*) para los nodos rama (los iconos de cerrado y abierto). En los nodos hoja se omita el segundo parámetro. En los nodos rama, si se omite *linkID2*, el icono se utiliza tanto para el estado abierto como para el cerrado.

Ejemplo

En el siguiente ejemplo se añaden dos nodos a una instancia de Tree denominada `my_tr` y se llama a la función `setIcon()` para especificar un icono en la biblioteca con el identificador de vinculación `imageIcon` para el segundo nodo.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr` y añadir un icono a la biblioteca con el identificador de vinculación `imageIcon`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
  - elemento de biblioteca con el identificador de vinculación imageIcon
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

// Establecer clip de película como icono del segundo nodo.
my_tr.setIcon(my_tr.getTreeNodeAt(1), "imageIcon");
```

Tree.setIsBranch()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Utilización

`treeInstance.setIsBranch(`*`node`*`, `*`isBranch`*`)`

Parámetros

`node` Nodo XML.

`isBranch` Valor booleano que indica si el nodo es una rama (`true`) o no (`false`).

Valor devuelto

Ninguno.

Descripción

Método; especifica si el nodo tiene un icono de carpeta y una flecha de ampliación y si tiene nodos secundarios o puede tenerlos. Un nodo se configura automáticamente como rama si tiene nodos secundarios; sólo tiene que llamar a `setIsBranch()` si desea crear una carpeta vacía. Puede crear ramas que aún no cuenten con nodos secundarios; por ejemplo, si desea que se carguen nodos secundarios sólo cuando un usuario abre una carpeta.

Al llamar a `setIsBranch()` se actualizan todas las vistas.

Ejemplo

En el siguiente ejemplo se añade un solo nodo a una instancia de Tree denominada `my_tr` y se llama a `setIsBranch()` para convertirla en una rama sin elementos secundarios.

Primero debe añadir una instancia del componente Tree al escenario y denominarla `my_tr`; a continuación, añada el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='Inbox' data='0'/>");
my_tr.dataProvider = trDP_xml;

// Establecer primero nodo como rama.
my_tr.setIsBranch(my_tr.getTreeNodeAt(0), true);
```

Tree.setIsOpen()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
treeInstance.setIsOpen(node, open [, animate [, fireEvent]])
```

Parámetros

node Nodo XML.

open Valor booleano que abre un nodo (`true`) o lo cierra (`false`).

animate Valor booleano que determina si la transición que se está abriendo está animada (`true`) o no (`false`). Este parámetro es opcional.

fireEvent Valor booleano que determina si los eventos `nodeOpen` y `nodeClose` se distribuyen (`true`) o no (`false`) cuando se abre o cierra el nodo del árbol. Este parámetro es opcional. El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Método; abre o cierra un nodo.

Ejemplo

En el siguiente ejemplo se crean dos nodos en una instancia de Tree denominada `my_tr` y se llama al método `setIsOpen()` para abrir el segundo nodo.

```
/**
 Se requiere:
  - Componente Tree en el escenario (nombre de instancia: my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Establecer segundo nodo como abierto.
my_tr.setIsOpen(my_tr.getTreeNodeAt(1), true);
```

CAPÍTULO 51

Clase Tween

51

Herencia (clase raíz)

Nombre de clase de ActionScript mx.transitions.Tween

La clase Tween permite utilizar ActionScript para mover fácilmente los clips de película, cambiar su tamaño y hacer que se desvanezcan en el escenario especificando una propiedad del clip de película de destino para crear una animación con interpolación durante varios fotogramas o segundos. La clase Tween también permite especificar diversos métodos de suavizado. El *suavizado* es la aceleración o desaceleración gradual durante una animación, que hace que las animaciones parezcan más realistas. Por ejemplo, las opciones de un componente de lista desplegable que cree podrían aumentar gradualmente su velocidad cerca del principio de una animación a medida que aparecen opciones y ralentizarse antes de que las opciones se detengan al final de la animación a medida que se extiende la lista. Flash proporciona muchos métodos de suavizado que contienen ecuaciones para esta aceleración y desaceleración, y que cambian la animación de suavizado de la forma correspondiente.

La clase Tween también invoca controladores de eventos de forma que el código pueda responder cuando una animación comienza, se detiene, reanuda o incrementa el valor de su propiedad de interpolación. Por ejemplo, puede iniciar una segunda animación interpolada cuando la primera interpolación invoca su controlador de eventos `Tween.onMotionStopped`, que indica que la primera interpolación se ha detenido.

Resumen de métodos de la clase Tween

En la tabla siguiente se enumeran los métodos de la clase Tween:

Método	Descripción
`Tween.yoyo()`	Indica a la animación interpolada que continúe desde su valor actual hasta un nuevo valor.
`Tween.fforward()`	Avanza la animación interpolada directamente al final de la animación.

Método	Descripción
`Tween.nextFrame()`	Avanza la animación interpolada al siguiente fotograma.
`Tween.prevFrame()`	Dirige la animación interpolada al fotograma anterior al actual.
`Tween.resume()`	Reanuda una animación interpolada a partir del punto en que se ha detenido en la animación.
`Tween.rewind()`	Retrocede una animación interpolada hasta el principio de la misma.
`Tween.start()`	Inicia la animación interpolada desde el principio.
`Tween.stop()`	Detiene la animación interpolada en su posición actual.
`Tween.toString()`	Devuelve el nombre de clase, "[Tween]".
`Tween.yoyo()`	Ordena a la animación interpolada que se reproduzca a la inversa desde el último sentido de incrementos de la propiedad de interpolación.

Resumen de propiedades de la clase Tween

En la tabla siguiente se enumeran las propiedades de la clase Tween.

Propiedad	Descripción
`Tween.duration`	Duración de la animación interpolada en fotogramas o segundos. Sólo lectura.
`Tween.finish`	Último valor interpolado del final de la animación interpolada. Sólo lectura.
`Tween.FPS`	Número de fotogramas por segundo de la animación interpolada. Sólo lectura.
`Tween.position`	Valor actual de la propiedad del clip de película de destino que se está interpolando. Sólo lectura.
`Tween.time`	Tiempo transcurrido de la duración de la animación. Sólo lectura.

Resumen del controlador de eventos de la clase Tween

En la tabla siguiente se enumeran los controladores de eventos de la clase Tween.

Evento	Descripción
`Tween.onMotionChanged`	Controlador de eventos; se invoca con cada cambio de la propiedad de interpolación del objeto que se está animando.
`Tween.onMotionFinished`	Controlador de eventos; se invoca cuando el objeto Tween finaliza su animación.
`Tween.onMotionResumed`	Controlador de eventos; se invoca cuando se llama al método `Tween.resume()`, que hace que se reanude la animación interpolada.
`Tween.onMotionStarted`	Controlador de eventos; se invoca cuando se llama al método `Tween.start()`, que hace que se inicie la animación interpolada.
`Tween.onMotionStopped`	Controlador de eventos; se invoca cuando se llama al método `Tween.stop()`, que hace que se detenga la animación interpolada.

Utilización de la clase Tween

Para utilizar los métodos y las propiedades de la clase Tween, debe utilizar el operador `new` para crear una nueva instancia de la clase. Por ejemplo, para aplicar en el escenario una instancia de una interpolación a un objeto de clip de película denominada `myMovieClip_mc`, debe usar el código siguiente para crear una nueva instancia de `mx.transitions.Tween`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(myMovieClip_mc, "_x",
  mx.transitions.easing.Elastic.easeOut, 0, 300, 3, true);
```

Parámetros de la clase Tween

Para crear una nueva instancia de una clase Tween es necesario pasar varios parámetros. Debe indicar el objeto de clip de película de destino, la propiedad del clip de película a la que va a afectar la interpolación, el rango en el que se va a interpolar el objeto y un método de suavizado que se va a utilizar para calcular el valor de la propiedad de interpolación.

El constructor de la clase mx.transitions.Tween tiene el siguiente nombre y tipos de parámetro:

```
Tween( obj:Object, prop:String, func:Function, begin:Number, finish:Number,
  duration:Number, useSeconds:Boolean )
```

obj Objeto de clip de película al que se va a aplicar la instancia de Tween.

prop Nombre de cadena de una propiedad de `obj` en la que se van a interpolar los valores.

func El método de suavizado que calcula un efecto de suavizado para los valores de propiedades del objeto interpolado. Véase "Métodos y clases de suavizado" en la página 1354.

begin Número que indica el valor inicial de `prop` (propiedad del objeto de destino que se va a interpolar).

finish Número que indica el valor de fin de `prop` (propiedad del objeto de destino que se va a interpolar).

duration Número que indica la duración del movimiento de interpolación. Si se omite, se establece una duración infinita de manera predeterminada.

useSeconds Valor booleano que indica que se utilizarán segundos si el valor especificado en el parámetro `duration` es `true` o fotogramas si el valor es `false`.

Métodos y clases de suavizado

Cuando se crea una instancia de la clase Tween, se utiliza el parámetro `func` para especificar una función o un método que proporcione un cálculo de suavizado. Flash proporciona cinco clases de suavizado, cada una con tres métodos que indican a qué parte del movimiento de transición se aplicará el efecto de suavizado: al principio de la animación, al final o en ambas partes. Se ofrece además, una clase de suavizado `None` con un método `easeNone` para indicar que no se puede utilizar ningún suavizado.

Las siguientes clases y componentes utilizan las clases y los métodos de suavizado:

- La clase mx.transitions.Tween para suavizar efectos en una animación interpolada.
- La clase mx.transitions.TransitionManager para suavizar efectos en las transiciones. Véase Capítulo 48, "Clase TransitionManager", en la página 1275.
- Algunos componentes de la versión 2 de la arquitectura de componentes de Macromedia. Véase "Aplicación de métodos de suavizado a componentes" en la página 1356.

Las seis clases de cálculo de suavizado se describen en la tabla siguiente:

Clase de suavizado	Descripción
Back	Extiende la animación cuando se supera el límite de la transición en uno o ambos extremos para producir el efecto de ser empujado hacia atrás desde el límite.
Bounce	Añade un efecto de rebote dentro del rango de la transición en uno o ambos extremos. El número de rebotes depende de la duración: cuanto mayor sea la duración, producirá un mayor número de rebotes.
Elastic	Añade un efecto elástico que está fuera del rango de la transición en uno o ambos extremos. La duración no afecta al grado de elasticidad.
Regular	Añade un movimiento más lento en uno o ambos extremos. Esta función permite añadir un efecto de aceleración, desaceleración o ambos.
Strong	Añade un movimiento más lento en uno o ambos extremos. Este efecto es similar al suavizado Regular, pero más pronunciado.
None	Añade un movimiento uniforme de principio a fin sin efectos, ralentización ni aceleración. Esta transición también se denomina transición lineal.

Cada una de estas seis clases de cálculo de suavizado tiene tres métodos de suavizado que indican la parte de la animación en la que se aplicará el efecto de suavizado. Además, la clase de suavizado None tiene un cuarto método de suavizado: easeNone. Los métodos de suavizado se describen en la tabla siguiente:

Método	Descripción
easeIn	Proporciona el efecto de suavizado al principio de la transición.
easeOut	Proporciona el efecto de suavizado al final de la transición.
easeInOut	Proporciona el efecto de suavizado al principio y al final de la transición.
easeNone	Indica que no se va a utilizar ningún cálculo de suavizado. Sólo se proporciona en la clase de suavizado None.

Aplicación de métodos de suavizado a componentes

Los diversos métodos de suavizado pueden aplicarse además a los componentes de la versión 2 de la arquitectura de componentes de Macromedia. Sólo pueden aplicarse los métodos de suavizado a los siguientes componentes de la versión 2: Accordion, ComboBox, DataGrid, List, Menu y Tree. Cada componente utiliza los métodos de suavizado para permitir distintas personalizaciones. Por ejemplo, los componentes Accordion, ComboBox y Tree permiten seleccionar una clase de suavizado para utilizar sus respectivas animaciones de apertura y cierre. Por el contrario, el componente Menu permite definir únicamente el número de milisegundos que dura la animación.

Aplicación de métodos de suavizado a un componente Accordion

En esta sección se describe cómo añadir un componente Accordion a un documento de Flash, cómo añadir unas cuantas diapositivas secundarias y cómo cambiar el método de suavizado y la duración predeterminados. Si decide utilizar este código en un proyecto, reduzca el valor de la propiedad `openDuration` para evitar molestar a los usuarios con animaciones que son demasiado lentas cuando se abren y cierre los paneles secundarios del componente Accordion.

Para aplicar un método de suavizado distinto al componente Accordion:

1. Cree un nuevo documento de Flash y guárdelo como accordion.fla.
2. Arrastre una copia del componente Accordion al escenario.
3. Abra el inspector de propiedades y escriba `my_acc` en el cuadro de texto Nombre de instancia.
4. Inserte una nueva capa sobre Capa 1 y denomínela *acciones*.
5. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones:

```
import mx.core.View;
import mx.transitions.easing.*;
my_acc.createChild(View, "studio_view", {label:"Studio"});
my_acc.createChild(View, "dreamweaver_view", {label:"Dreamweaver"});
my_acc.createChild(View, "flash_view", {label:"Flash"});
my_acc.createChild(View, "coldfusion_view", {label:"ColdFusion"});
my_acc.createChild(View, "contribute_view", {label:"Contribute"});
my_acc.setStyle("openEasing", Bounce.easeOut);
my_acc.setStyle("openDuration", 3500);
```

Este código importa las clases de suavizado, de forma que puede escribir `Bounce.easeOut` en lugar de hacer referencia a cada una de las clases con nombres completos como `mx.transitions.easing.Bounce.easeOut`. A continuación, el código añade cinco nuevos paneles secundarios al componente Accordion (Studio, Dreamweaver, Flash, ColdFusion y Contribute). Las últimas dos líneas de código establecen el estilo de suavizado, desde el método de suavizado predeterminado hasta `Bounce.easeOut`, y establecen la duración de la animación en 3.500 milisegundos (3,5 segundos).

6. Guarde el documento y seleccione Control > Probar película para obtener una previsualización dinámica del archivo en el entorno de prueba.

   Haga clic en cada una de las distintas barras de encabezado (título) para ver las animaciones modificadas y pasar de un panel a otro.

   Si desea aumentar la velocidad de la animación, especifique un valor de `openDuration` inferior a 3.500 milisegundos. La duración predeterminada de la animación es de 250 milisegundos (un cuarto de segundo).

Aplicación de métodos de suavizado a un componente ComboBox

El proceso para cambiar el método de suavizado predeterminado en un componente ComboBox es similar al ejemplo de "Aplicación de métodos de suavizado a un componente Accordion" en la página 1356, en el que se ha modificado la animación del componente Accordion. En el siguiente ejemplo, se utiliza ActionScript para añadir dinámicamente el componente al escenario en tiempo de ejecución.

Para aplicar métodos de suavizado a un componente ComboBox:

1. Cree un nuevo documento de Flash y guárdelo como combobox.fla.
2. Arrastre una copia del componente ComboBox desde el panel Componentes a la biblioteca del documento actual.

   NOTA El componente aparece en la biblioteca (no en el escenario) y está disponible para el archivo SWF en tiempo de ejecución.

3. Inserte una capa y asígnele el nombre acciones.

   Asegúrese de que la capa acciones está por encima de la Capa 1.

4. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones:

```
import mx.transitions.easing.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 20);
var product_array:Array = new Array("Studio", "Dreamweaver", "Flash",
  "ColdFusion", "Contribute", "Breeze", "Director", "Flex");
my_cb.dataProvider = product_array;
my_cb.move(10, 10);
my_cb.setSize(140, 22);
my_cb.setStyle("openDuration", 2000);
my_cb.setStyle("openEasing", Elastic.easeOut);
```

Después de importar cada uno de los métodos de suavizado, lo que sucede en la primera línea de código, el método `createClassObject()` crea una instancia del componente ComboBox. La palabra clave `this` en la segunda línea de código hace referencia a la línea de tiempo principal del archivo SWF. Esta línea de código coloca el componente en el escenario en tiempo de ejecución y le asigna el nombre de instancia `my_cb`.

A continuación, cree una matriz denominada `product_array` que contenga una lista del software de Macromedia. Utilice esta matriz en la siguiente línea de código para establecer la propiedad `dataProvider` en la matriz de nombres de producto. A continuación, utilice el método `setSize()` para cambiar el tamaño de la instancia de componente, establezca el valor de `openDuration` en 2.000 milisegundos (2 segundos) y cambie el método de suavizado por `Elastic.easeOut`.

NOTA

Al igual que en los ejemplos anteriores, se importan las clases de suavizado, que permiten utilizar una versión corta del nombre de clase en lugar del nombre completo de `mx.transitions.easing.Elastic.easeOut`.

5. Guarde el documento actual y seleccione Control > Probar película para ver el documento en el entorno de prueba.
6. Haga clic en el componente ComboBox en el escenario para utilizar la clase de suavizado especificada a fin de animar la lista desplegable de nombres de productos.

NOTA

Utilice un método de suavizado como Elastic o Bounce para los componentes ComboBox o Accordion. Algunos usuarios se distraerán si las opciones están en movimiento durante mucho tiempo antes de que puedan leerlas y seleccionarlas del menú. Pruebe las aplicaciones y configuraciones individuales y decida si los métodos de suavizado mejoran el documento de Flash o si más bien lo empeoran.

Animación del componente DataGrid

Flash 8 también permite retocar ligeramente las animaciones que se utilizan para seleccionar los elementos de un componente (por ejemplo, DataGrid, Tree, ComboBox o List). Aunque las animaciones apenas se perciben, en algunos casos resulta conveniente controlar los pequeños detalles o aumentar la velocidad de la animación.

Para añadir suavizado al componente DataGrid:

1. Cree un nuevo documento de Flash y guárdelo como datagrid.fla.
2. Arrastre una instancia del componente DataGrid al escenario y asígnele el nombre de instancia `my_dg`.
3. Inserte una capa y asígnele el nombre acciones.

   Asegúrese de que la capa acciones está por encima de la Capa 1.
4. Añada el siguiente código ActionScript a la capa acciones:

```
import mx.transitions.easing.*;
my_dg.setSize(320, 240);
my_dg.addColumn("product");
my_dg.getColumnAt(0).width = 304;
my_dg.rowHeight = 60;
my_dg.addItem({product:"Studio"});
my_dg.addItem({product:"Dreamweaver"});
my_dg.addItem({product:"Flash"});
my_dg.setStyle("selectionEasing", Elastic.easeInOut);
my_dg.setStyle("selectionDuration", 1000);
```

   Este código ActionScript importa las clases de suavizado y cambia el tamaño de la instancia del componente en el escenario a 320 píxeles (anchura) por 240 píxeles (altura). A continuación, cree una columna con el nombre *producto* y cambie el tamaño de la columna a 304 píxeles (anchura). La propia cuadrícula de datos tiene una anchura de 320 píxeles, aunque la barra de desplazamiento tiene una anchura de 16 píxeles, lo que produce una diferencia de 304 píxeles. A continuación se establece la altura de la fila en 60 píxeles, lo hace que las animaciones de suavizado sean más fáciles de ver.

   Las siguientes tres líneas de código ActionScript añaden elementos a la instancia de cuadrícula para poder hacer clic y ver las animaciones. Finalmente, se establecen las propiedades `selectionEasing` y `selectionDuration` mediante el método `setStyle()`. El método de suavizado se establece en `Elastic.easeInOut`, con una duración (`duration`) de 1.000 milisegundos (un segundo, que es cinco veces más largo que el valor predeterminado de 200 milisegundos).

5. Guarde el documento y seleccione Control > Probar película para ver el resultado en el entorno de prueba.

   Cuando haga clic en un elemento de la instancia de DataGrid, verá la aceleración y desaceleración de la selección mediante el efecto elástico. Debería verse fácilmente la animación porque se ha aumentado significativamente la duración.

NOTA	También puede utilizar las mismas propiedades (`selectionEasing` y `selectionDuration`) con los componentes ComboBox, List y Tree.

Tween.continueTo()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.continueTo(finish, duration)
```

Parámetros

`finish` Número que indica el valor de fin de la propiedad del objeto de destino que se va a interpolar.

`duration` Número que indica la duración temporal o número de fotogramas para el movimiento de interpolación; se mide en duración temporal si el parámetro `useSeconds` de `Tween.start()` se establece en true o en fotogramas si se establece en false. Para más información sobre el parámetro `useSeconds`, consulte `Tween.start()` en la página 1375.

Valor devuelto

Ninguno.

Descripción

Método; indica a la animación interpolada que continúe la interpolación a partir de su punto de animación actual, hasta un nuevo punto de duración y finalización.

Ejemplo

En este ejemplo, el evento `onMotionFinished` activa un controlador, que indica a la instancia de Tween que continúe su animación hasta unos nuevos valores de finalización (`finish`) y duración (`duration`), mediante una llamada al método `Tween.continueTo()`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Elastic.easeOut,0, 200, 3, true);
myTween.onMotionFinished = function() {
  var myFinish:Number = 100;
  var myDuration:Number = 5;
  myTween.continueTo( myFinish, myDuration );
};
```

Tween.duration

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.duration

Descripción

Propiedad (sólo lectura); número que indica la duración de la animación interpolada en fotogramas o segundos. Esta propiedad se establece como un parámetro cuando se crea una nueva instancia de Tween o cuando se llama al método `Tween.yoyo()`.

Ejemplo

En el ejemplo siguiente se realiza un seguimiento del valor actual de `duration` de un objeto Tween obteniendo el valor de la propiedad `Tween.duration`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height, 50, false);
var theDuration:Number = myTween.duration;
trace(theDuration);
```

Tween.fforward()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.fforward()

Valor devuelto

Ninguno.

Descripción

Método; avanza la animación interpolada directamente a su valor final.

Ejemplo

En este ejemplo, se llama a `Tween.fforward()` para avanzar una animación con interpolación directamente a su valor final, tras lo cual se activa de inmediato el evento `onMotionFinished`. Un controlador para el evento `Tween.onMotionFinished` llama a `Tween.yoyo()`. Por lo tanto, la animación interpolada se inicia de manera visible con el efecto inverso del método `Tween.yoyo()`, ya que se omitió la animación inicial hasta su valor final. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width - img1_mc._width, 8,
  true);

myTween.fforward();

myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

Tween.finish

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.finish

Descripción

Propiedad (sólo lectura); número que indica el valor de fin de la propiedad del objeto de destino que se va a interpolar. Esta propiedad se establece como un parámetro cuando se crea una nueva instancia de Tween o cuando se llama al método `Tween.yoyo()`.

Ejemplo

El siguiente ejemplo devuelve el valor actual de `finish` de una instancia de Tween. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img1_mc._height,
  50, false);
var myFinish:Number = myTween.finish;
trace(myFinish);
```

Tween.FPS

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.FPS

Descripción

Propiedad; número de fotogramas por segundo calculado en la animación interpolada. De manera predeterminada, la velocidad de fotogramas actual del escenario se utiliza para calcular la animación interpolada. Al establecer esta propiedad se recalcula el número de incrementos en la propiedad animada que se muestra cada segundo como el valor de la propiedad `Tween.FPS` en lugar de la velocidad de fotogramas actual del escenario. Al establecer el valor de la propiedad `Tween.FPS`, la velocidad de fotogramas real del escenario no cambia.

NOTA	La propiedad Tween.FPS devuelve undefined a menos que primero se establezca explícitamente.

Ejemplo

En el ejemplo siguiente se crean dos animaciones interpoladas establecidas en dos valores distintos de `FPS`. Se muestran los valores actuales de `FPS` para ambas instancias de Tween. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc` y otra instancia de clip de película denominada `img2_mc`:

```
import mx.transitions.Tween;
var myTween1:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img1_mc._height,
  400, false);
myTween1.FPS = 1;
var myFPS1:Number = myTween1.FPS;
trace("myTween1.FPS:" + myFPS1);

var myTween2:Tween = new Tween(img2_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img2_mc._height,
  400, false);
myTween2.FPS = 12;
var myFPS2:Number = myTween2.FPS;
trace("myTween2.FPS:" + myFPS2);
```

Tween.nextFrame()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.nextFrame()

Valor devuelto

Ninguno.

Descripción

Método; avanza la animación interpolada al siguiente fotograma de una animación que se ha detenido. Utilice este método para avanzar fotograma a fotograma en una animación interpolada después de utilizar el método `Tween.stop()` para detenerlo.

NOTA	Este método sólo puede utilizarse en interpolaciones basadas en fotogramas. Para especificar que una interpolación se basa en fotogramas, debe establecer el valor false en el parámetro `useSeconds`.

Ejemplo

Este ejemplo aplica una animación interpolada al clip de película `img1_mc`. La animación se reproduce indefinidamente a partir de su punto inicial mediante una llamada al método `Tween.start()` desde un controlador activado por el evento `Tween.onMotionFinished`. Al hacer clic en un botón denominado forwardByFrame_btn se llama al método `Tween.stop()` para detener la animación y a continuación se llama al método `Tween.nextFrame()`. Si se hace clic en el botón durante la animación con interpolación, se detiene la animación y se avanza solamente un fotograma. Al crear la instancia de Tween, el parámetro `useSeconds` se declara como false para que la interpolación se base en fotogramas. Este proceso es necesario para utilizar el método `Tween.nextFrame()`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 60, false);

myTween.onMotionFinished = function() {
  myTween.start();
};

forwardByFrame_btn.onRelease = function() {
   myTween.stop();
   myTween.nextFrame();
};
```

Tween.onMotionChanged

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.onMotionChanged = function() {
  // ...
};
```

Descripción

Controlador de eventos; se invoca con cada cambio de la propiedad de interpolación del objeto que se está animando. La gestión de este evento permite la reacción del código cuando se incrementa en un valor la propiedad del clip de película de destino que se interpola.

Ejemplo

En este ejemplo se aplica una interpolación al clip de película `img1_mc`. Con cada incremento de la interpolación en la propiedad `_x` del clip de película, se invoca el controlador de eventos `onMotionChanged` y se muestra un mensaje de seguimiento que indica la nueva posición del clip de película interpolado. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width-img1_mc._width, 3,
  true);

myTween.onMotionChanged = function() {
  trace( this.position );
};
```

Tween.onMotionFinished

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.onMotionFinished = function() {
  // ...
};
```

Descripción

Controlador de eventos; se invoca cuando la animación llega al final de su duración. La gestión de este evento permite la reacción del código en el punto en que finaliza la animación con interpolación.

Ejemplo

En el siguiente ejemplo se aplica una interpolación al clip de película `img1_mc`. Cuando la interpolación llega al final de su animación, invoca el controlador de eventos `onMotionFinished` que llama al método `Tween.yoyo()`. Por lo tanto, la interpolación puede completar su animación antes de llamar al método `Tween.yoyo()` para invertir la animación. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width-img1_mc._width, 3,
  true);
myTween.FPS = 30;
myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

Tween.onMotionResumed

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.onMotionResumed = function() {
  // ...
};
```

Descripción

Controlador de eventos; se invoca cuando se llama al método `Tween.resume()`. La gestión de este evento permite la reacción del código en el punto en que se reanuda la animación con interpolación.

Ejemplo

El siguiente ejemplo aplica una animación interpolada al clip de película `img1_mc`. Se repite la animación de forma que se reproduzca repetidamente desde el punto inicial llamando al método `Tween.start()` desde un controlador de eventos `onMotionFinished`. Al hacer clic en un botón denominado `stopTween_btn`, se llama al método `Tween.stop()` para detener la animación interpolada en su valor actual. Al hacer clic en un botón denominado resumeTween_btn, se llama el método `Tween.resume()` para reanudar la animación interpolada desde el punto en el que se ha detenido. Cuando se llama al método `Tween.resume()`, la instancia de Tween invoca el controlador `onMotionResumed`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`, una instancia de clip de película denominada `stopTween_btn` y otra instancia de clip de película denominada `resumeTween_btn`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 3, true);

myTween.onMotionFinished = function() {
  myTween.start();
};

myTween.onMotionResumed = function() {
  trace("onMotionResumed");
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};

resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.onMotionStarted

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.onMotionStarted = function() {
  // ...
};
```

Descripción

Controlador de eventos; se invoca cuando se ha iniciado nuevamente la animación en el transcurso de la misma o después de haber finalizado. Este controlador de eventos no se invoca en el primer inicio de una animación interpolada. Si se llama al método `Tween.start()`, `Tween.yoyo()` o `Tween.yoyo()` para reiniciar una animación finalizada o reiniciarla durante una animación, se invoca el controlador de eventos `onMotionStarted`. La gestión de este evento permite la reacción del código en el punto en que se reinicia la animación con interpolación en algún momento después de haberse iniciado ya.

Ejemplo

El siguiente ejemplo aplica una animación interpolada al clip de película `img1_mc`. Se repite la animación de forma que se reproduzca repetidamente desde el punto inicial llamando al método `Tween.start()` desde un controlador de eventos `Tween.onMotionFinished`. Cuando se llama al método `Tween.start()`, la instancia de Tween invoca el controlador `Tween.onMotionStarted`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 4, true);

myTween.onMotionFinished = function() {
  myTween.start();
};

myTween.onMotionStarted = function() {
  trace("onMotionStarted");
};
```

Tween.onMotionStopped

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.onMotionStopped = function() {
  // ...
};
```

Descripción

Controlador de eventos; se difunde cuando la animación interpolada finaliza por completo o cuando se llama al método `Tween.stop()`. La gestión de este evento permite la reacción del código en el punto en que se detiene la animación con interpolación.

Ejemplo

El siguiente ejemplo aplica una animación interpolada al clip de película `img1_mc`. Cuando la instancia de Tween finaliza la animación, invoca el controlador de eventos `Tween.onMotionStopped`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width-img1_mc._width, 3,
  true);

myTween.onMotionStopped = function() {
  trace("onMotionStopped");
};
```

Tween.position

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.position

Descripción

Propiedad (sólo lectura); el valor actual de la propiedad del objeto de destino que se está interpolando. Este valor se actualiza con cada fotograma dibujado de la animación interpolada.

Ejemplo

En el ejemplo siguiente se realiza un seguimiento de los valores actuales de `Tween.position` y `Tween.position` del objeto Tween con los que la posición de interpolación finaliza en el último fotograma de la animación interpolada. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new mx.transitions.Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, 0, Stage.width-img1_mc._width, 3,
  true);
myTween.onMotionChanged = function() {
  var myPosition:Number = myTween.position;
  var myFinish:Number = myTween.finish;
  trace(myPosition + " : " + myFinish);
};
```

Tween.prevFrame()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.prevFrame()

Valor devuelto

Ninguno.

Descripción

Método; reproduce el fotograma anterior de la animación interpolada desde el punto en el que se ha detenido en una animación detenida. Utilice este método para retroceder la animación interpolada un fotograma hasta un momento después de haberla detenido mediante el método `Tween.stop()`.

NOTA	Este método sólo puede utilizarse en interpolaciones basadas en fotogramas. Para especificar que una interpolación se basa en fotogramas, debe establecer el valor false en el parámetro `Tween.start()useSeconds`. Para más información sobre el parámetro `useSeconds`, consulte `Tween.start()` en la página 1375.

Ejemplo

Este ejemplo aplica una animación interpolada al clip de película `img1_mc`. La animación se reproduce indefinidamente a partir de su punto inicial mediante una llamada al método `Tween.start()` desde un controlador activado por el evento `onMotionFinished`. Al hacer clic en un botón denominado forwardByFrame_btn se llama al método `Tween.stop()` para detener la animación y a continuación se llama al método `Tween.prevFrame()`. Al hacer clic en el botón durante la animación interpolada, se detiene la animación y después se invierte en un solo fotograma. Al hacer clic en el botón resumeTween_btn, se llama al método Tween.resume() y la animación interpolada se reanuda. Al crear la instancia de Tween, el parámetro `useSeconds` se declara como `false` para que la interpolación se base en fotogramas. Este proceso es necesario para utilizar el método `Tween.nextFrame()`. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`, una instancia de clip de película denominada `resumeTween_btn` y otra instancia de clip de película denominada `reverseByFrame_btn`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, -img1_mc._width, Stage.width, 50,
  false);

myTween.onMotionFinished = function() {
  myTween.start();
};

reverseByFrame_btn.onRelease = function() {
   myTween.stop();
   myTween.prevFrame();
};
resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.resume()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.resume()

Valor devuelto

Ninguno.

Descripción

Método; reanuda la reproducción de una animación interpolada que se ha detenido. Utilice este método para continuar una animación interpolada después de haberla detenido mediante el método Tween.stop().

NOTA	Este método sólo puede utilizarse en interpolaciones basadas en fotogramas. Para especificar que una interpolación se basa en fotogramas, debe establecer el valor false en el parámetro useSeconds.

Ejemplo

Este ejemplo aplica una animación interpolada al clip de película img1_mc. La animación se reproduce indefinidamente a partir de su punto inicial mediante una llamada al método Tween.start() desde un controlador activado por el evento onMotionFinished. Al hacer clic en el botón stopTween_btn, se llama el método Tween.stop() para detener la animación interpolada en su valor actual. Al hacer clic en el botón resumeTween_btn, se llama al método Tween.resume() para reanudar la animación interpolada desde el punto en el que se ha detenido. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada img1_mc, una instancia de clip de película denominada stopTween_btn y otra instancia de clip de película denominada resumeTween_btn:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, -img1_mc._width, Stage.width, 3,
  true);
```

```
myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};
resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.rewind()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.rewind()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; mueve la reproducción de una animación interpolada a su valor inicial. Si se llama a `Tween.rewind()` mientras se está reproduciendo la animación interpolada, la animación se rebobina hasta su valor inicial y continúa reproduciéndose. Si se llama a `Tween.rewind()` mientras la animación interpolada está detenida o ha finalizado, la animación interpolada se rebobina hasta su valor inicial y permanece detenida. Utilice este método para rebobinar una animación interpolada hasta su punto inicial después de haberla detenido mediante el método `Tween.stop()` o para rebobinar una animación interpolada mientras se reproduce.

Ejemplo

El siguiente ejemplo aplica una animación interpolada al clip de película `img1_mc`. La animación se reproduce indefinidamente a partir de su punto inicial mediante una llamada al método `Tween.start()` desde un controlador activado por el evento `Tween.onMotionFinished`. Al hacer clic en el botón rewindTween_btn, se llama al método `Tween.rewind()` para rebobinar la animación interpolada a su punto inicial. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`, una instancia de clip de película denominada `stopTween_btn`, una instancia de clip de película denominada `rewindTween_btn` y una instancia de clip de película denominada `resumeTween_btn`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, img1_mc._width, Stage.width, 8,
  true);

myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
  myTween.stop();
};

rewindTween_btn.onRelease = function() {
  myTween.rewind();
};

resumeTween_btn.onRelease = function() {
  myTween.resume();
};
```

Tween.start()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.start()

Valor devuelto

Ninguno.

Descripción

Método; inicia la reproducción de una animación interpolada a partir de su punto inicial. Este método se utiliza para reiniciar una instancia de Tween desde el principio de su animación después de que se haya detenido o completado su animación.

Ejemplo

Este ejemplo crea un nuevo objeto Tween que anima la propiedad `_x` del clip de película `img1_mc`. Después de que la animación interpolada finalice y llame al controlador de eventos `Tween.onMotionFinished`, se vuelve a reproducir la interpolación llamando al método `Tween.start()`. El resultado es un clip de película que se mueve por el escenario de izquierda a derecha y después inicia la animación de nuevo cuando llega al final del escenario. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 50, false);

myTween.onMotionFinished = function() {
  myTween.start();
};
```

Tween.stop()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`tweenInstance``.stop()`

Valor devuelto

Ninguno.

Descripción

Método; detiene la reproducción de una animación interpolada en su valor actual.

Ejemplo

En el siguiente ejemplo se aplica una animación interpolada a la propiedad `_x` del clip de película `img1_mc`. El controlador `onRelease()` del clip de película `stopTween_btn` llama al método `Tween.stop()` para detener la animación interpolada y un controlador `onRelease()` del clip de película `resumeTween_btn` llama al método `Tween.resume()` para reanudar la animación desde una posición de animación detenida. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`, una instancia de clip de película denominada `stopTween_btn` y otra instancia de clip de película denominada `resumeTween_btn`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, img1_mc._width, Stage.width, 8,
  true);

myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};

resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.time

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
tweenInstance.time
```

Descripción

Propiedad (sólo lectura); número actual de segundos transcurridos de la duración de la animación si el parámetro `useSeconds` se estableció en `true` al crear la instancia de `Tween`. Si el parámetro `useSeconds` de la animación se estableció en `false`, `Tween.time` devuelve el número actual de fotogramas pasados a la animación del objeto Tween.

Ejemplo

El siguiente ejemplo devuelve el valor de time de una instancia de Tween. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new mx.transitions.Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, 0, Stage.width-img1_mc._width, 10,
  false);
myTween.onMotionChanged = function() {
  var myCurrentTime:Number = myTween.time;
  var myCurrentDuration:Number = myTween.duration;
trace(myCurrentTime + " of " + myCurrentDuration);
};
```

Tween.toString()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`tweenInstance``.toString()`

Valor devuelto

Se devuelve la siguiente cadena: "[Tween]".

Descripción

Método; devuelve el nombre de clase, "[Tween]".

Ejemplo

En el siguiente ejemplo se identifica un objeto Tween llamando al método `Tween.toString()` para devolver "`[Tween]`", que identifica el nombre de clase del objeto. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_alpha",
  mx.transitions.easing.None.easeNone,0, 100, 50, false);
var theClassName:String = myTween.toString();
trace(theClassName);
```

Tween.yoyo()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

tweenInstance.yoyo()

Valor devuelto

Ninguno.

Descripción

Método; ordena a la animación interpolada que se reproduzca a la inversa desde el último sentido de incrementos de la propiedad de interpolación. Si se llama a este método antes de que una animación del objeto Tween haya finalizado, la animación salta bruscamente al final de su reproducción y a continuación se reproduce en sentido inverso desde ese punto. Puede lograr un efecto de una animación que está completando su reproducción y después invertir toda la reproducción llamando al método `Tween.yoyo()` en un controlador de eventos `Tween.onMotionFinished`. Este proceso garantiza que el efecto inverso del método Tween.yoyo no se iniciará hasta que la animación interpolada actual se haya completado. Véase `Tween.onMotionFinished` en la página 1366.

Ejemplo

En el siguiente ejemplo se utiliza el evento `Tween.onMotionFinished` para activar un controlador y se ordena a la instancia de Tween que anime el clip de película `img1_mc` en sentido inverso llamando al método `Tween.yoyo()`. El resultado es un clip de película que se mueve de izquierda a derecha en el escenario y después invierte el sentido, moviéndose de derecha a izquierda en un bucle de animación. Para este ejemplo se requiere en el escenario una instancia de clip de película denominada `img1_mc`:

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 4, true);
myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

CAPÍTULO 52

Clase UIComponent

52

La clase UIComponent no representa un componente visual; contiene métodos, propiedades y eventos que permiten a los componentes de Macromedia compartir algunos comportamientos comunes. Todos los componentes de la versión 2 de la arquitectura de componentes de Macromedia amplían UIComponent. La clase UIComponent permite realizar lo siguiente:

- Recibir selecciones y entradas del teclado
- Activar y desactivar componentes
- Cambiar el tamaño mediante el diseño

Para utilizar los métodos y propiedades de UIComponent, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar a `UIComponent.setFocus()` desde el componente RadioButton, deberá escribir el código siguiente:

```
myRadioButton.setFocus();
```

Sólo necesita crear una instancia de UIComponent si está utilizando la versión 2 de la arquitectura de componentes de Macromedia para crear un componente nuevo. Incluso en tal caso, UIComponent se crea de forma implícita por medio de otras subclases, como por ejemplo Button. Si necesita crear una instancia de UIComponent, utilice el código siguiente:

```
class MyComponent extends mx.core.UIComponent;
```

Clase UIComponent (API)

Herencia MovieClip > Clase UIObject > UIComponent

Nombre de clase de ActionScript mx.core.UIComponent

Los métodos, propiedades y eventos de la clase UIComponent permiten controlar el comportamiento común de los componentes visuales de Flash.

Resumen de métodos de la clase UIComponent

En la siguiente tabla se enumeran los métodos de la clase UIComponent.

Método	Descripción
`UIComponent.getFocus()`	Devuelve una referencia al objeto seleccionado.
`UIComponent.setFocus()`	Define la selección en la instancia de componente.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase UIComponent de la clase UIObject. Al llamar a estos métodos desde el objeto UIComponent, debe utilizarse la forma *`UIComponentInstance.methodName`*.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Resumen de propiedades de la clase UIComponent

En la siguiente tabla se enumeran las propiedades de la clase UIComponent.

Propiedad	Descripción
`UIComponent.enabled`	Indica si el componente puede recibir selecciones y entradas.
`UIComponent.tabIndex`	Número que indica el orden de tabulación para un componente de un documento.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase UIComponent de la clase UIObject. Al acceder a estas propiedades desde el objeto UIComponent, debe utilizarse la forma *`UIComponentInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.right`	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
`UIObject.width`	Anchura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.x`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
`UIObject.y`	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Resumen de eventos de la clase UIComponent

En la siguiente tabla se enumeran los eventos de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase UIComponent de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

UIComponent.enabled

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.enabled
```

Descripción

Propiedad; indica si el componente puede aceptar selecciones y entradas del ratón (`true`) o no (`false`). El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se establece la propiedad `enabled` de un componente CheckBox en `false`:

```
checkBoxInstance.enabled = false;
```

UIComponent.focusIn

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.focusIn = function(eventObj:Object) {
  //...
};
componentInstance.addEventListener("focusIn", listenerObject);
```

Sintaxis 2:

```
on (focusIn) {
  // ...
}
```

Descripción

Evento; notifica al detector que el objeto se ha seleccionado con el teclado.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `focusIn`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

El siguiente código desactiva un componente Button, `btn`, mientras un usuario escribe en el componente TextInput, `txt`, y habilita el botón cuando el usuario hace clic en él:

```
var txt:mx.controls.TextInput;
var btn:mx.controls.Button;

var txtListener:Object = new Object();
txtListener.focusOut = function() {
  _root.btn.enabled = true;
}
txt.addEventListener("focusOut", txtListener);

var txtListener2:Object = new Object();
txtListener2.focusIn = function() {
  _root.btn.enabled = false;
}
txt.addEventListener("focusIn", txtListener2);
```

Véase también

`EventDispatcher.addEventListener()`, `UIComponent.focusOut`

UIComponent.focusOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
on(focusOut){
   ...
}
listenerObject = new Object();
listenerObject.focusOut = function(eventObject){
   ...
}
componentInstance.addEventListener("focusOut", listenerObject)
```

Descripción

Evento; notifica a los detectores que el objeto ya no tiene la selección del teclado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

El segundo ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector. Una instancia de componente (*`componentInstance`*) distribuye un evento (en este caso, `focusOut`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

El siguiente código desactiva un componente Button, `btn`, mientras un usuario escribe en el componente TextInput, `txt`, y habilita el botón cuando el usuario hace clic en él:

```
var txt:mx.controls.TextInput;
var btn:mx.controls.Button;

var txtListener:Object = new Object();
txtListener.focusOut = function() {
  _root.btn.enabled = true;
}
txt.addEventListener("focusOut", txtListener);

var txtListener2:Object = new Object();
txtListener2.focusIn = function() {
  _root.btn.enabled = false;
}
txt.addEventListener("focusIn", txtListener2);
```

Véase también

`EventDispatcher.addEventListener()`, `UIComponent.focusIn`

UIComponent.getFocus()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.getFocus();
```

Parámetros

Ninguno.

Valor devuelto

Referencia al objeto que recibe actualmente la selección.

Descripción

Método; devuelve una referencia al objeto que recibe la selección del teclado.

Ejemplo

El código siguiente devuelve una referencia al objeto que recibe la selección que asigna a la variable `tmp` :

```
var tmp = checkbox.getFocus();
```

UIComponent.keyDown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
on(keyDown){
   ...
}
listenerObject = new Object();
listenerObject.keyDown = function(eventObject){
   ...
}
componentInstance.addEventListener("keyDown", listenerObject)
```

Descripción

Evento; notifica a los detectores que se ha presionado una tecla. Se trata de un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

El segundo ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector. Una instancia de componente (*`componentInstance`*) distribuye un evento (en este caso, `keyDown`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

Ejemplo

El código siguiente hace que un icono parpadee cuando se presiona una tecla:

```
formListener.handleEvent = function(eventObj)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

UIComponent.keyUp

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
on(keyUp){
  ...
}
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
  ...
}
componentInstance.addEventListener("keyUp", listenerObject)
```

Descripción

Evento; notifica a los detectores cuando se suelta una tecla. Se trata de un evento de nivel bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

El segundo ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `keyUp`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

El código siguiente hace que un icono parpadee cuando se suelta una tecla:

```
formListener.handleEvent = function(eventObj)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

UIComponent.setFocus()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setFocus();
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; define la selección en esta instancia del componente. La instancia con la selección recibe todas las entradas del teclado.

Ejemplo

El código siguiente selecciona la instancia `checkbox`:

```
checkbox.setFocus();
```

UIComponent.tabIndex

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
instance.tabIndex
```

Descripción

Propiedad; número que indica el orden de tabulación para un componente en un documento.

Ejemplo

El código siguiente define el valor de `tmp` en la propiedad `tabIndex` de la instancia `checkbox`:

```
var tmp = checkbox.tabIndex;
```

CAPÍTULO 53

Clase UIEventDispatcher

53

Nombre de clase de ActionScript mx.events.UIEventDispatcher

Herencia Clase EventDispatcher > UIEventDispatcher

La clase UIEventDispatcher se mezcla con la clase UIComponent y permite que los componentes emitan ciertos eventos.

Si se desea que un objeto que no se hereda de UIComponent distribuya ciertos eventos, se puede utilizar UIEventDispatcher.

Resumen de métodos de la clase UIEventDispatcher

En la tabla siguiente se muestra el método de la clase UIEventDispatcher.

Método	Descripción
`UIEventDispatcher.removeEventListener()`	Elimina un detector registrado de una instancia de componente. Este método sustituye al método `eventDispatcher.removeEventListenter()`.

Métodos heredados de la clase EventDispatcher

En la tabla siguiente se enumeran los métodos que hereda la clase UIEventDispatcher de la clase EventDispatcher. Al llamar a estos métodos desde el objeto UIEventDispatcher, debe utilizarse la forma *`UIEventDispatcherInstance.methodName`*.

Método	Descripción
`EventDispatcher.addEventListener()`	Registra un detector para una instancia de componente.
`EventDispatcher.dispatchEvent()`	Distribuye un evento a todos los detectores registrados.

Resumen de eventos de la clase UIEventDispatcher

En la siguiente tabla se enumeran los eventos de la clase UIEventDispatcher.

Método	Descripción
UIEventDispatcher.keyDown	Se difunde cuando se presiona una tecla.
UIEventDispatcher.keyUp	Se difunde cuando se suelta una tecla presionada.
UIEventDispatcher.load	Se difunde cuando un componente se carga en Flash Player.
UIEventDispatcher.mouseDown	Se difunde cuando se presiona el ratón.
UIEventDispatcher.mouseOut	Se difunde cuando el ratón se mueve fuera de una instancia de componente.
UIEventDispatcher.mouseOver	Se difunde cuando el ratón se mueve sobre una instancia de componente.
UIEventDispatcher.mouseUp	Se difunde cuando se presiona y se suelta el botón del ratón.
UIEventDispatcher.unload	Se difunde cuando un componente se descarga de Flash Player.

UIEventDispatcher.keyDown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.keyDown = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("keyDown", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se presiona una tecla y la aplicación Flash está seleccionada.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.keyUp

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("keyUp", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se suelta una tecla que estaba presionada y la aplicación Flash está seleccionada.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.load

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.load = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("load", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se carga un componente en Flash Player.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.mouseDown

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.mouseDown = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("mouseDown", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando una aplicación Flash está seleccionada y se presiona el ratón.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.mouseOut

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.mouseOut = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("mouseOut", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando una aplicación Flash está seleccionada y el ratón se mueve fuera de una instancia de componente.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.mouseOver

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.mouseOver = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("mouseOver", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando una aplicación Flash está seleccionada y el ratón se mueve sobre una instancia de componente.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

UIEventDispatcher.mouseUp

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.mouseUp = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("mouseUp", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando una aplicación Flash está seleccionada y se presiona y se suelta el botón del ratón.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Para más información, consulte "Clase EventDispatcher" en la página 515.

UIEventDispatcher.removeEventListener()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
componentInstance.removeEventListener(event, listener)
```

Parámetros

event Cadena que es el nombre del evento.

listener Referencia a un objeto detector o a una función.

Valor devuelto

Ninguno.

Descripción

Método; quita el registro de un objeto detector desde una instancia de componente que difunde un evento. Este método sustituye al evento `EventDispatcher.removeEventListener()` que se encuentra en la clase base EventDispatcher.

UIEventDispatcher.unload

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.unload = function(eventObject){
  // Introducir aquí el código propio.
}
componentInstance.addEventListener("unload", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se descarga un componente de Flash Player.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento.

CAPÍTULO 54

Clase UIObject

54

Herencia MovieClip > UIObject

Nombre de clase de ActionScript mx.core.UIObject

UIObject es la clase base para todos los componentes de la versión 2 de la arquitectura de componentes de Macromedia; no es un componente visual. La clase UIObject ajusta el objeto MovieClip de ActionScript y contiene funciones y propiedades que permiten a los componentes de la versión 2 compartir algunos comportamientos comunes. El ajuste de la clase MovieClip permite a Macromedia añadir nuevos eventos y ampliar funciones en el futuro sin que ello afecte al contenido. Asimismo, el ajuste de la clase MovieClip permitirá a los usuarios que no conocen los conceptos tradicionales de Flash como "película" y "fotograma" utilizar las propiedades, métodos y eventos para crear aplicaciones basadas en componentes sin conocer dichos conceptos.

La clase UIObject incorpora lo siguiente:

- Estilos
- Eventos
- Cambio del tamaño mediante la escala

Para utilizar los métodos y propiedades de la clase UIObject, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar al método `UIObject.setSize()` desde el componente RadioButton, deberá escribir el código siguiente:

```
myRadioButton.setSize(30, 30);
```

Sólo necesita crear una instancia de UIObject si está utilizando la versión 2 de la arquitectura de componentes de Macromedia para crear un componente nuevo. Incluso en este caso, UIObject se crea con frecuencia de forma implícita por medio de otras subclases, por ejemplo Button. Si necesita crear una instancia de UIObject, utilice el código siguiente:

```
class MyComponent extends UIObject;
```

Resumen de métodos de la clase UIObject

En la siguiente tabla se enumeran los métodos de la clase UIObject.

Método	Descripción
`UIObject.createClassObject()`	Crea un objeto en la clase especificada.
`UIObject.createLabel()`	Crea un subobjeto TextField para utilizarlo durante la creación de componentes.
`UIObject.createObject()`	Crea un subobjeto en un objeto.
`UIObject.destroyObject()`	Elimina una instancia de componente.
`UIObject.doLater()`	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
`UIObject.getStyle()`	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
`UIObject.invalidate()`	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
`UIObject.move()`	Mueve el objeto a la posición indicada.
`UIObject.redraw()`	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
`UIObject.setSize()`	Cambia el tamaño del objeto al indicado.
`UIObject.setSkin()`	Define un aspecto en el objeto.
`UIObject.setStyle()`	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Resumen de propiedades de la clase UIObject

En la siguiente tabla se enumeran las propiedades de la clase UIObject.

Propiedad	Descripción
`UIObject.bottom`	Posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente. Sólo lectura.
`UIObject.height`	Altura del objeto, expresada en píxeles. Sólo lectura.
`UIObject.left`	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.

Propiedad	Descripción
UIObject.right	Posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Sólo lectura.
UIObject.scaleX	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Posición del borde superior del objeto con respecto a su elemento principal correspondiente. Sólo lectura.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Anchura del objeto, expresada en píxeles. Sólo lectura.
UIObject.x	Borde izquierdo del objeto, expresado en píxeles. Sólo lectura.
UIObject.y	Borde superior del objeto, expresado en píxeles. Sólo lectura.

Resumen de eventos de la clase UIObject

En la siguiente tabla se enumeran los eventos de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

UIObject.bottom

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.bottom
```

Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición inferior del objeto con respecto a la parte inferior de su elemento principal correspondiente. Para definir esta propiedad, llame a `UIObject.move()`.

Ejemplo

En este ejemplo se desplaza la casilla de verificación para alinearla debajo del borde inferior del cuadro de lista:

```
myCheckbox.move(myCheckbox.x, form.height - listbox.bottom);
```

UIObject.createClassObject()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.createClassObject(className, instanceName, depth,
  initObject)
```

Parámetros

`className` Objeto que indica la clase de la nueva instancia.

`instanceName` Cadena que indica el nombre de instancia de la instancia nueva.

`depth` Número que indica la profundidad de la nueva instancia.

`initObject` Objeto que contiene propiedades de inicialización para la nueva instancia.

Valor devuelto

Objeto `UIObject` que es una instancia de la clase especificada.

Descripción

Método; crea una instancia de un componente en tiempo de ejecución. Utilice la sentencia `import` y especifique el nombre del paquete de clase antes de llamar a este método. Además, el componente debe estar en la biblioteca del archivo FLA.

Ejemplo

El siguiente código importa los activos del componente Button y, a continuación, crea un subobjeto del componente Button:

```
import mx.controls.Button;
createClassObject(Button,"button2",5,{label:"Test Button"});
```

En el ejemplo siguiente se crea un objeto `CheckBox`:

```
import mx.controls.CheckBox;
form.createClassObject(CheckBox, "cb", 0, {label:"Check this"});
```

También se puede especificar el nombre del paquete de clase mediante la siguiente sintaxis:

```
createClassObject(mx.controls.Button, "button2", 5, {label:"Test Button"});
```

UIObject.createLabel()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
createLabel(name, depth, text)
```

Parámetros

`name` Cadena que especifica el nombre de instancia.

`depth` Número que indica la profundidad de la nueva instancia.

`text` Texto de la etiqueta.

Valor devuelto

Un objeto `TextField`.

Descripción

Método; crea un subobjeto `TextField`. La mayoría de los componentes lo utilizan para obtener un objeto de texto ligero para mostrar texto en el componente, al tiempo que heredan los métodos y propiedades de estilo y cambio de tamaño del componente. Este método se utiliza para crear nuevos componentes. El objeto TextField que se crea es el mismo objeto `TextField` que se crea mediante `MovieClip.createTextField()`, pero tiene la ventaja añadida de que hereda métodos y propiedades útiles del UIObject principal.

Un objeto TextField que utilice `UIObject.createLabel()` para crearse en un componente puede aprovechar los siguientes métodos heredados de UIObject para establecer el cambio de tamaño y los estilos en el contexto del UIObject principal:

- TextField.getPreferredHeight() : Number
- TextField.getPreferredWidth() : Number
- TextField.setStyle(styleName : String, value)
- TextField.setSize(width : Number, height : Number)
- TextField.setValue(text : String)

NOTA

Los objetos TextField creados con `UIObject.createLabel()` tienen una propiedad `TextField._visible` inicial establecida en `false`. Esta propiedad se utiliza para evitar las posibles oscilaciones mientras el componente principal llama a `UIObject.setSize()`. La propiedad `TextField._visible` se establece en true cuando se llama a `UIObject.draw()` después de cambiar el tamaño de los objetos secundarios del componente principal.

Para más información, consulte el archivo de ejemplo MultilineCell.as en "Ejemplo sencillo de procesador de celdas" en la página 116.

Ejemplo

El siguiente ejemplo crea una instancia de TextField denominada `multiLineLabel` en el método `UIComponent.createChildren()` de un componente:

```
public function createChildren():Void {
  var myTextField_txt:TextField = this.createLabel("multiLineLabel", 900,
  "Hello World");
  // Establecer el atributo de estilo fontSize de TextField.
  myTextField_txt.setStyle("fontSize", 18);
  // Establecer el tamaño inicial de TextField.
  myTextField_txt.setSize(myTextField_txt.getPreferredWidth(),
  myTextField_txt.getPreferredHeight());
  // Establecer la ubicación inicial de TextField en el centro del
  escenario.
  myTextField_txt._x = (Stage.width/2) - (myTextField_txt._width/2);
  myTextField_txt._y = (Stage.height/2) - (myTextField_txt._height/2);
}
```

UIObject.createObject()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.createObject(linkageName, instanceName, depth,
  initObject)
```

Parámetros

linkageName Cadena que indica el identificador de vinculación de un símbolo en la biblioteca.

instanceName Cadena que indica el nombre de instancia de la instancia nueva.

depth Número que indica la profundidad de la nueva instancia.

initObject Objeto que contiene propiedades de inicialización para la nueva instancia.

Valor devuelto

Objeto UIObject que es una instancia del símbolo.

Descripción

Método; crea un subobjeto en un objeto. Este método suelen utilizarlo sólo los desarrolladores de componentes o los desarrolladores avanzados.

Ejemplo

En el ejemplo siguiente se crea una instancia CheckBox en el objeto `form`:

```
form.createObject("CheckBox", "sym1", 0);
```

UIObject.destroyObject()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
destroyObject(instanceName)
```

Parámetros

instanceName Cadena que indica el nombre de instancia del objeto que se va a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina una instancia de componente.

Ejemplo

En el siguiente ejemplo se elimina la primera instancia `my_ti` de TextInput cuando se hace clic en el botón. Con un componente Button y un componente TextInput en la biblioteca del documento actual, añada el siguiente código al primer fotograma de la línea de tiempo principal:

```
//Crear instancias de TextInput y Button
this.createClassObject(mx.controls.TextInput, "my_ti", 1, {text:"Hello
  World"});
this.createClassObject(mx.controls.Button, "my_button", 2, {label:"My
  Button"});
//Mover botón debajo de entrada de texto
my_button.move(my_ti.left, Stage.height - my_ti.bottom);

//Crear objeto detector para clic de botón
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object){
  destroyObject("my_ti");
}
//Añadir detector
my_button.addEventListener("click", buttonListener);
```

UIObject.doLater()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.doLater(target, "function")
```

Parámetros

target Referencia a la línea de tiempo que contiene la función especificada.

function Cadena que indica el nombre de una función a la que hay que llamar después de que haya pasado un fotograma en el clip de película del componente (para que estén disponibles las propiedades establecidas en el inspector de componentes o en el inspector de propiedades).

Valor devuelto

Ninguno.

Descripción

Método; llama a una función definida por el usuario solamente después de que el componente haya terminado de establecer todas sus propiedades a partir del inspector de propiedades o el inspector de componentes. Todos los componentes de la versión 2 que heredan de UIObject tienen el método `doLater()`.

Es posible que las propiedades de componentes establecidas en el inspector de propiedades o el inspector de componentes no se encuentren disponibles de forma inmediata para ActionScript en la línea de tiempo. Por ejemplo, al intentar rastrear la propiedad `label` de un componente CheckBox mediante ActionScript, en el primer fotograma del archivo SWF se produce un error que no se notifica, aunque el componente aparezca en el escenario según lo previsto.

Aunque las propiedades establecidas en una clase o en un script de fotograma están disponibles de forma inmediata, la mayoría de las propiedades asignadas en el inspector de propiedades o en el inspector de componentes no se establecen hasta el siguiente fotograma en el propio componente.

Si bien cualquier enfoque que retrase el acceso a la propiedad resolvería este problema, la solución más simple y directa es utilizar el método `doLater()`.

Ejemplo

En el siguiente ejemplo se muestra la utilización del método `doLater()`:

```
// se llama a doLater() desde la instancia de componente

myCheckBox.doLater(this, "delay");

// Función o método al que se llama desde doLater().

function delay() {
  trace(myCheckBox.label); // Se puede rastrear la propiedad ahora
  // aquí va cualquier sentencia adicional
}
```

UIObject.draw

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.draw = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("draw", listenerObject);
```

Sintaxis 2:

```
on (draw) {
  // ...
}
```

Descripción

Evento; notifica a los detectores que el objeto está a punto de dibujar sus gráficos. Se trata de un evento de nivel bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `draw`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

El código siguiente vuelve a dibujar el objeto `form2` cuando se ha dibujado el objeto `form`:

```
formListener.draw = function(eventObj:Object) {
  form2.redraw(true);
}
form.addEventListener("draw", formListener);
```

Véase también

`EventDispatcher.addEventListener()`

UIObject.getStyle()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.getStyle(*propertyName*)

Parámetros

propertyName Cadena que indica el nombre de la propiedad de estilo (por ejemplo, `"fontWeight"`, `"borderStyle"` y así sucesivamente).

Valor devuelto

Valor de la propiedad de estilo. El valor puede ser cualquier tipo de datos.

Descripción

Método; obtiene la propiedad de estilo de la declaración de estilo o del objeto. Si la propiedad de estilo es un estilo heredado, los antecesores del objeto pueden ser el origen del valor de estilo.

Para ver una lista de los estilos admitidos por cada componente, consulte las entradas individuales de componente. Véase también el "Utilización de un estilo global y estilos personalizados y de clase en un mismo documento" en *Utilización de componentes*.

Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia `ib` en negrita cuando la propiedad de estilo `fontWeight` de la instancia `cb` sea negrita:

```
if (cb.getStyle("fontWeight") == "bold") {
   ib.setStyle("fontWeight", "bold");
};
```

UIObject.height

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.height
```

Descripción

Propiedad (sólo lectura); número que indica la altura del objeto, expresada en píxeles. Para cambiar la propiedad `height`, llame a `UIObject.setSize()`.

Ejemplo

En el ejemplo siguiente se aumenta la altura de la casilla de verificación:

```
myCheckbox.setSize(myCheckbox.width, myCheckbox.height + 10);
```

UIObject.hide

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.hide = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("hide", listenerObject);
```

Sintaxis 2:

```
on (hide) {
  // ...
}
```

Descripción

Evento; se difunde cuando la propiedad `visible` del objeto cambia de `true` a `false`.

Ejemplo

El controlador siguiente muestra un mensaje en el panel Salida cuando el objeto con el que está asociado queda oculto.

```
on (hide) {
  trace("I've become invisible.");
}
```

Véase también

UIObject.reveal

UIObject.invalidate()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.invalidate()

Valor devuelto

Ninguno.

Descripción

Método; marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.

Este método es principalmente útil para desarrolladores de componentes personalizados nuevos. Un componente personalizado es probable que admita un número de operaciones que cambien el aspecto del componente.

Con frecuencia, la mejor forma de crear un componente es concentrando la lógica para actualizar el aspecto del componente en el método `draw()`. Si el componente tiene un método `draw()`, se puede llamar a `invalidate()` en el componente para volver a dibujarlo. Para más información sobre la definición de un método `draw()`, consulte "Definición del método draw()" en *Utilización de componentes*.

Todas las operaciones que cambian el aspecto del componente pueden llamar a `invalidate()` en lugar de volver a dibujar el componente. Esto tiene algunas ventajas: el código no se duplica innecesariamente y se pueden agrupar varios cambios en un redibujado en lugar de provocar varios y repetidos redibujados.

Ejemplo

En el ejemplo siguiente se marca la instancia `pBar` de ProgressBar para volver a dibujarla:

```
pBar.invalidate();
```

UIObject.left

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.left

Descripción

Propiedad (sólo lectura); número que indica en píxeles el borde izquierdo del objeto con respecto a su elemento principal correspondiente. Para definir esta propiedad, llame a `UIObject.move()`.

UIObject.load

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.load = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("load", listenerObject);
```

Sintaxis 2:

```
on (load) {
  //...
}
```

Descripción

Evento; notifica a los detectores que se ha creado el subobjeto de este objeto.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `load`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

En el ejemplo siguiente se crea una instancia de MySymbol después de cargar la instancia `form`:

```
var formListener:Object = new Object();
formListener.load = function(eventObj:Object) {
  form.createObject("MySymbol", "sym1", 0);
};
form.addEventListener("load", formListener);
```

UIObject.move

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.move = function(eventObject:Object):Void {
  // ...
};
componentInstance.addEventListener("move", listenerObject);
```

Sintaxis 2:

```
on (move) {
  // ...
}
```

Descripción

Evento; notifica a los detectores que el objeto se ha movido.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `move`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

En el siguiente ejemplo se llama al método `move()` para mover un componente Button, `my_button`, desde su posición actual hasta la esquina superior izquierda (10,10) del escenario:

```
var my_button:mx.controls.Button;
my_button.addEventListener("move", doMove);
function doMove(evt_obj:Object):Void {
  trace(evt_obj.target + " moved from {oldX:" + evt_obj.oldX + ", oldY:" +
  evt_obj.oldY + "} to {x:" + evt_obj.target.x + ", y:" + evt_obj.target.y
  + "}");
}
my_button.move(10, 10);
```

UIObject.move()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.move(*x*, *y*, *noEvent*)

Parámetros

x Número que indica la posición de la esquina superior izquierda del objeto con respecto a su elemento principal correspondiente.

y Número que indica la posición de la esquina superior izquierda del objeto con respecto a su elemento principal correspondiente.

noEvent Valor booleano que indica si debe distribuirse el evento move.

Valor devuelto

Ninguno.

Descripción

Método; mueve el objeto hasta la posición indicada. Pase sólo valores enteros a `UIObject.move()` o, de lo contrario, el componente puede aparecer borroso.

Ejemplo

En el ejemplo siguiente, la casilla de verificación se desplaza hacia la derecha 10 píxeles:

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

En el siguiente ejemplo se llama al método `move()` para mover un componente Button, `my_button`, desde su posición actual hasta la esquina superior izquierda (10,10) del escenario:

```
var my_button:mx.controls.Button;
my_button.addEventListener("move", doMove);
function doMove(evt_obj:Object):Void {
  trace(evt_obj.target + " moved from {oldX:" + evt_obj.oldX + ", oldY:" +
  evt_obj.oldY + "} to {x:" + evt_obj.target.x + ", y:" + evt_obj.target.y
  + "}");
}
my_button.move(10, 10);
```

UIObject.redraw()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.redraw(*always*)

Parámetros

always Valor booleano. Si es `true`, el método dibuja el objeto aunque no se haya llamado a `invalidate()`. Si es `false`, el método dibuja el objeto sólo si se ha llamado a `invalidate()`.

Valor devuelto

Ninguno.

Descripción

Método; fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.

Ejemplo

En el ejemplo siguiente se crean una casilla de verificación y un botón y se dibujan ya que no se espera que otros scripts modifiquen el formulario:

```
form.createClassObject(mx.controls.CheckBox, "cb", 0);
form.createClassObject(mx.controls.Button, "b", 1);
form.redraw(true)
```

UIObject.resize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("resize", listenerObject);
```

Sintaxis 2:

```
on (resize) {
  // ...
}
```

Descripción

Evento; notifica a los detectores que se ha cambiado el tamaño de un objeto.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `resize`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

En el ejemplo siguiente se llama al método `setSize()` para que `sym1` sea la mitad de ancho y un cuarto de alto cuando `form` se mueva:

```
var formListener:Object = new Object();
formListener.resize = function(eventObj:Object):Void {
  form.sym1.setSize(sym1.width / 2, sym1.height / 4);
};
form.addEventListener("resize", formListener);
```

En el siguiente ejemplo se llama al método `setSize()` para cambiar el tamaño de un componente Button, `my_button`, a 200 píxeles de ancho por 100 píxeles de alto:

```
var my_button:mx.controls.Button;

my_button.addEventListener("resize", doSize);
function doSize(evt_obj:Object):Void {
  trace(evt_obj.target + " resized from {oldWidth:" + evt_obj.oldWidth + ",
  oldHeight:" + evt_obj.oldHeight + "} to {width:" + evt_obj.target.width +
  ", height:" + evt_obj.target.height + "}");
}
my_button.setSize(200, 100);
```

UIObject.reveal

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.reveal = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("reveal", listenerObject);
```

Sintaxis 2:

```
on (reveal) {
  // ...
}
```

Descripción

Evento; se difunde cuando la propiedad `visible` del objeto cambia de `false` a `true`.

Ejemplo

El controlador siguiente muestra un mensaje en el panel Salida cuando el objeto con el que está asociado aparece en pantalla.

```
on (reveal) {
  trace("I've become visible.");
}
```

Véase también

`UIObject.hide`

UIObject.right

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.right

Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente. Para definir esta propiedad, llame a `UIObject.move()`.

Ejemplo

En el ejemplo siguiente se desplaza la casilla de verificación para alinearla debajo del borde derecho del cuadro de lista:

```
myCheckbox.move(form.width - listbox.right, myCheckbox.y);
```

UIObject.scaleX

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.scaleX

Descripción

Propiedad; número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su anchura y se define la variable `tmp` en el factor de escala horizontal:

```
checkbox.scaleX = 200;
var tmp:Number = checkbox.scaleX;
```

UIObject.scaleY

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.scaleY

Descripción

Propiedad; número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.

Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su altura y se define la variable `tmp` en el factor de escala vertical:

```
checkbox.scaleY = 200;
var tmp:Number = checkbox.scaleY;
```

UIObject.setSize()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.setSize(*width*, *height*, *noEvent*)

Parámetros

width Número que indica la anchura del objeto, expresada en píxeles.

height Número que indica la altura del objeto, expresada en píxeles.

noEvent Valor booleano que indica si debe distribuirse el evento resize.

Valor devuelto

Ninguno.

Descripción

Método; cambia el tamaño del objeto por el tamaño especificado. Debe pasar sólo valores enteros a `UIObject.setSize()` o, de lo contrario, el componente puede aparecer borroso. Este método (y todos los métodos y propiedades de UIObject) está disponible desde cualquier instancia de componente.

La llamada a este método desde una instancia de ComboBox hace que se ajuste el tamaño del cuadro combinado y cambie la propiedad `rowHeight` de la lista que contiene.

NOTA	Algunos componentes sólo permiten modificar las dimensiones de altura *o* anchura. Por ejemplo, los componentes CheckBox y RadioButton no permiten modificar la altura.

Ejemplo

En este ejemplo se cambia el tamaño de la instancia del componente `pBar` a 100 píxeles de ancho y 100 píxeles de alto:

```
pBar.setSize(100, 100);
```

En el siguiente ejemplo se llama al método `setSize()` para cambiar el tamaño de un componente Button, `my_button`, a 200 píxeles de ancho por 100 píxeles de alto:

```
var my_button:mx.controls.Button;

my_button.addEventListener("resize", doSize);
function doSize(evt_obj:Object):Void {
  trace(evt_obj.target + " resized from {oldWidth:" + evt_obj.oldWidth + ",
  oldHeight:" + evt_obj.oldHeight + "} to {width:" + evt_obj.target.width +
  ", height:" + evt_obj.target.height + "}");
}
my_button.setSize(200, 100);
```

UIObject.setSkin()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.setSkin(*id*, *linkageName*)

Parámetros

id Número que indica la profundidad del aspecto de componente.

linkageName Cadena que indica un elemento de la biblioteca.

Valor devuelto

Referencia al clip de película (aspecto) que se asoció.

Descripción

Método; define un aspecto en la instancia de componente. Utilice este método en un archivo de clase de componente cuando cree un componente. Para más información, consulte "Asignación de aspectos" en *Utilización de componentes*.

No se puede utilizar este método para definir los aspectos de un componente en tiempo de ejecución (por ejemplo, la forma en la que se definen los estilos de un componente en tiempo de ejecución).

Ejemplo

Este ejemplo es un fragmento de código del archivo de clase de un componente nuevo llamado Shape. Crea una variable `themeShape` y la define en el identificador de vinculación del aspecto. En el método `createChildren()`, se llama al método `setSkin()` y se pasa el id `1` y la variable que aloja el identificador de vinculación del aspecto:

```
class Shape extends UIComponent {

  static var symbolName:String = "Shape";
  static var symbolOwner:Object = Shape;
  var className:String = "Shape";

  var themeShape:String = "circle_skin"

  function Shape() {
  }

  function init(Void):Void {
    super.init();
  }

  function createChildren():Void {
    setSkin(1, themeShape);
    super.createChildren();
  }
}
```

UIObject.setStyle()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.setStyle(*propertyName*, *value*)

Parámetros

propertyName Cadena que indica el nombre de la propiedad de estilo. Los estilos compatibles dependerán del componente. Cada componente permite establecer un conjunto de estilos distinto. Por ejemplo, "Personalización del componente TextArea" en la página 1218 muestra una tabla de estilos, incluido `fontWeight`. Así pues, en un componente TextArea, puede utilizar `fontWeight` como valor del parámetro *propertyName*.

value Valor de la propiedad. Si el valor es una cadena, debe ir entrecomillado.

Valor devuelto

Ninguno.

Descripción

Método; define la propiedad de estilo en la declaración de estilos o en el objeto. Si la propiedad de estilo es un estilo heredado, la notificación del nuevo valor se enviará a los elementos secundarios.

Para ver una lista de estilos admitidos por cada componente, consulte las entradas individuales de componente. Por ejemplo, los estilos del componente Button se enumeran en "Utilización de estilos con el componente Button" en la página 96.

Para mejorar el rendimiento puede cambiar estilos antes de cargarlos, calcularlos y aplicarlos a los objetos del archivo SWF. Si cambia los estilos antes de cargarlos y calcularlos, no tendrá que llamar a `setStyle`.

Macromedia recomienda establecer las propiedades en cada objeto porque se crean instancias de los objetos para mejorar el rendimiento cuando se utilizan estilos. Cuando añada dinámicamente instancias al escenario, establezca las propiedades en el parámetro `initObj` en la llamada que realice a `UIObject.createClassObject()`, como muestra el siguiente código ActionScript:

```
createClassObject(ComponentClass, "myInstance", 0, {styleName:"myStyle",
  color:0x99CCFF});
```

NOTA

En este ejemplo se utiliza la declaración de estilo personalizado myStyle. Para cambiar varias propiedades o cambiar propiedades de varias instancias de componente, cree una declaración de estilo personalizado. Flash representa con mayor rapidez un componente que utilice una declaración de estilo personalizado que uno que utilice `UIObject.setStyle()` para varias propiedades. Para más información, consulte "Definición de estilos personalizados para grupos de componentes" en *Utilización de componentes*.

Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia de la casilla de verificación `cb` en negrita:

```
cb.setStyle("fontWeight", "bold");
```

UIObject.top

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.top

Descripción

Propiedad (sólo lectura); número que indica en píxeles el borde superior del objeto con respecto a su elemento principal correspondiente. Para definir esta propiedad, llame a `UIObject.move()`.

UIObject.unload

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.unload = function(eventObject:Object):Void {
  // ...
};
componentInstance.addEventListener("unload", listenerObject);
```

Sintaxis 2:

```
on (unload) {
  // ...
}
```

Descripción

Evento; notifica a los detectores que se están descargando los subobjetos de este objeto.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `unload`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. Cada objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de componente.

Ejemplo

En el ejemplo siguiente se elimina `sym1` cuando se activa el evento `unload`:

```
function doUnload():Void {
  form.destroyObject(sym1);
}
form.addEventListener("unload", doUnload);
```

UIObject.visible

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.visible

Descripción

Propiedad; valor booleano que indica si el objeto es visible (`true`) o no (`false`).

Ejemplo

En el ejemplo siguiente se hace que la instancia de Loader `myLoader` sea visible:

```
myLoader.visible = true;
```

UIObject.width

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.width

Descripción

Propiedad (sólo lectura); número que indica la anchura del objeto, expresada en píxeles. Para cambiar la anchura, llame a `UIObject.setSize()`.

Ejemplo

En el ejemplo siguiente se aumenta la anchura de la casilla de verificación:

```
myCheckbox.setSize(myCheckbox.width + 10, myCheckbox.height);
```

UIObject.x

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.x

Descripción

Propiedad (sólo lectura); número que indica el borde izquierdo del objeto, expresado en píxeles. Para definir esta propiedad, llame a `UIObject.move()`.

Ejemplo

En el ejemplo siguiente, la casilla de verificación se desplaza hacia la derecha 10 píxeles:

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

UIObject.y

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

componentInstance.y

Descripción

Propiedad (sólo lectura); número que indica el borde superior del objeto, expresado en píxeles. Para definir esta propiedad, llame a `UIObject.move()`.

Ejemplo

En el ejemplo siguiente la casilla de verificación se desplaza hacia abajo 10 píxeles:

```
myCheckbox.move(myCheckbox.x, myCheckbox.y + 10);
```

CAPÍTULO 55

Componente UIScrollBar

55

El componente UIScrollBar permite añadir una barra de desplazamiento a un campo de texto Puede añadir una barra de desplazamiento a un campo de texto durante la edición o en tiempo de ejecución con ActionScript.

El componente UIScrollBar funciona como cualquier otra barra de desplazamiento. Contiene botones de flecha en ambos extremos y una guía de desplazamiento y un cuadro de desplazamiento (deslizador) en medio. Puede asociarse con cualquier borde de un campo de texto y utilizarse tanto vertical como horizontalmente.

Utilización del componente UIScrollBar

Para utilizar el componente UIScrollBar, compruebe que está activado el ajuste a objetos (Ver > Ajustar > Ajustar a objetos). A continuación, cree un campo de introducción de texto en el escenario y arrastre el componente UIScrollBar del panel Componentes a cualquier cuadrante del recuadro de delimitación del campo de texto.

Si la barra de desplazamiento es más corta que el tamaño combinado de sus flechas de desplazamiento, no se mostrará correctamente. Uno de los botones de flecha quedará oculto detrás del otro. Flash no proporciona funciones de comprobación de errores para estos casos. Es aconsejable ocultar la barra de desplazamiento con ActionScript. Si se cambia el tamaño de la barra de desplazamiento de forma que no haya espacio suficiente para el cuadro de desplazamiento (deslizador), Flash oculta el cuadro de desplazamiento.

A diferencia de muchos otros componentes, el componente UIScrollBar puede recibir entradas continuas del ratón como, por ejemplo, cuando el usuario mantiene presionado el botón del ratón en lugar de tener que hacer clic repetidas veces.

El componente UIScrollBar no permite la interacción con el teclado.

Parámetros de UIScrollBar

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente UIScrollBar en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

_targetInstanceName indica el nombre de instancia del campo de texto asociado con el componente UIScrollBar.

horizontal indica si la barra de desplazamiento está orientada horizontalmente (`true`) o verticalmente (`false`). El valor predeterminado es `false`.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente UIScrollBar en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

Es posible escribir ActionScript para controlar éstas y otras opciones adicionales para un componente UIScrollBar mediante sus propiedades, métodos y eventos. Para más información, consulte "Clase UIScrollBar" en la página 1437.

Creación de aplicaciones con el componente UIScrollBar

El siguiente procedimiento explica cómo añadir un componente UIScrollBar a una aplicación durante la edición.

Para crear una aplicación con el componente UIScrollBar:

1. Cree un campo de texto dinámico y asígnele el nombre de instancia **myText** en el inspector de propiedades.
2. En el inspector de propiedades, establezca el Tipo de línea del campo de introducción de texto en Multilínea o en Multilínea sin ajuste si tiene previsto utilizar la barra de desplazamiento horizontalmente.

3. En el Fotograma 1, utilice ActionScript para añadir texto suficiente al campo de forma que los usuarios deban desplazarse para poder verlo en su totalidad. Puede escribir el siguiente código:

```
myText.text="When the moon is in the seventh house and Jupiter aligns
  with Mars, then peace will guide the planet and love will rule the
  stars."
```

NOTA	Asegúrese de que el campo de texto en el escenario es lo suficientemente pequeño como para tener que desplazarse por él para poder ver todo el texto. Si no lo es, la barra de desplazamiento no aparece o puede aparecer simplemente como dos líneas sin control deslizador, que es la parte que se arrastra para desplazarse por el contenido.

4. Compruebe que está activado el ajuste a objetos (Ver > Ajustar > Ajustar a objetos).
5. Arrastre una instancia de UIScrollBar del panel Componentes al campo de introducción de texto junto al lado donde desea asociarlo. El componente debe solaparse con el campo de texto al soltar el ratón, para que quede correctamente vinculado al campo.

   La propiedad `_targetInstanceName` del componente se llena automáticamente con el nombre de instancia del campo de texto en el inspector de propiedades y el inspector de componentes. Si no aparece en la ficha Parámetros, es posible que no haya superpuesto suficientemente la instancia de UIScrollBar.
6. Seleccione Control > Probar película.

   La aplicación se ejecuta y la barra de desplazamiento desplaza el contenido al campo de texto.

También se puede crear una instancia del componente UIScrollBar y asociarla con un campo de texto en tiempo de ejecución mediante ActionScript.

El código siguiente crea una instancia de UIScrollBar orientada verticalmente y la asocia con el lado derecho de una instancia de campo de texto denominada **my_txt** y establece el tamaño de la barra de desplazamiento para ajustarla al tamaño del campo de texto:

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/
// Crear campo de texto.
this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Establecer el campo de texto de destino de la barra de desplazamiento.
my_sb.setScrollTarget(my_txt);
```

```
// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Personalización del componente UIScrollBar

El componente UIScrollBar puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Sin embargo, un componente UIScrollBar vertical no permite modificar la anchura y un componente UIScrollBar horizontal no permite modificar la altura. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o cualquier método o propiedad aplicable de la clase UIScrollBar.

NOTA

Si utiliza el método `UIObject.setSize()`, sólo puede cambiar la altura o la anchura de la instancia en función de que se trate de una barra de desplazamiento horizontal o vertical. Por tanto, el método `setSize()` omite el parámetro `height` o `width`.

No obstante, tenga en cuenta que, con el tema Halo, una barra de desplazamiento orientada verticalmente debe tener una anchura de 16 píxeles y una barra de desplazamiento orientada horizontalmente debe tener una altura de 16 píxeles. Estas dimensiones se determinan de manera estricta por el tema que se utilice actualmente con la barra de desplazamiento. Sólo es posible cambiar la dimensión de longitud de una barra de desplazamiento.

Es posible personalizar la apariencia de una instancia de UIScrollBar mediante el uso de estilos y aspectos.

Utilización de estilos con el componente UIScrollBar

El componente UIScrollBar admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`scrollTrackColor`	Sample	Color de fondo de la guía de desplazamiento. El valor predeterminado es 0xCCCCCC (gris claro).
`symbolColor`	Sample	Color de las flechas de desplazamiento arriba y abajo. El valor predeterminado es 0x000000 (negro).
`symbolDisabledColor`	Sample	Color de las flechas de desplazamiento arriba y abajo en una barra de desplazamiento desactivada. El valor predeterminado es 0x848384 (gris oscuro).

Utilización de aspectos con el componente UIScrollBar

El componente UIScrollBar utiliza 13 aspectos para la guía, el cuadro de desplazamiento (deslizador) y los botones. Para personalizar estos elementos de aspecto, edite los símbolos en la carpeta Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/States. Para más información, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*.

Las barras de desplazamiento horizontal y vertical utilizan los mismos aspectos verticales; cuando se muestra una barra de desplazamiento horizontal, el componente UIScrollBar gira los aspectos del modo correspondiente.

El componente UIScrollBar admite las siguientes propiedades de aspecto:

Propiedad	Descripción
`upArrowUpName`	Estado arriba (normal) de los botones arriba e izquierdo. El valor predeterminado es `ScrollUpArrowUp`.
`upArrowOverName`	Estado de los botones arriba e izquierdo cuando se desplaza el puntero sobre ellos. El valor predeterminado es `ScrollUpArrowOver`.
`upArrowDownName`	Estado de los botones arriba e izquierdo cuando se presionan. El valor predeterminado es `ScrollUpArrowDown`.
`downArrowUpName`	Estado arriba (normal) de los botones abajo y derecho. El valor predeterminado es `ScrollDownArrowUp`.

Propiedad	Descripción
`downArrowOverName`	Estado de los botones abajo y derecho cuando se desplaza el puntero sobre ellos. El valor predeterminado es `ScrollDownArrowOver`.
`downArrowDownName`	Estado de los botones abajo y derecho cuando se presionan. El valor predeterminado es `ScrollDownArrowDown`.
`scrollTrackName`	Símbolo que se utiliza para la guía de la barra de desplazamiento (fondo). El valor predeterminado es `ScrollTrack`.
`scrollTrackOverName`	Símbolo que se utiliza para la guía de desplazamiento (fondo) cuando se desplaza el puntero sobre ella. El valor predeterminado es `undefined`.
`scrollTrackDownName`	Símbolo que se utiliza para la guía de desplazamiento (fondo) cuando se presiona. El valor predeterminado es `undefined`.
`thumbTopName`	Extremos superior e izquierdo del cuadro de desplazamiento (deslizador). El valor predeterminado es `ScrollThumbTopUp`.
`thumbMiddleName`	Parte central (ampliable) del deslizador. El valor predeterminado es `ScrollThumbMiddleUp`.
`thumbBottomName`	Extremos inferior y derecho del deslizador. El valor predeterminado es `ScrollThumbBottomUp`.
`thumbGripName`	Control que se visualiza delante del deslizador. El valor predeterminado es `ScrollThumbGripUp`.

En el siguiente ejemplo se muestra cómo insertar una fina línea blanca en medio de la guía de desplazamiento.

Para crear símbolos de clip de película para aspectos de UIScrollBar:

1. Cree un nuevo archivo FLA.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y, a continuación, seleccione el archivo HaloTheme.fla.

   Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación. Para saber cuál es la ubicación exacta en el sistema operativo, consulte "Temas" en *Utilización de componentes*.
3. En el panel Biblioteca del tema, expanda la carpeta Flash UI Components 2/Themes/MMDefault y arrastre la carpeta ScrollBar Assets a la biblioteca del documento.
4. Expanda la carpeta ScrollBar Assets/States en la biblioteca del documento.
5. Abra los símbolos que desea personalizar.

   Por ejemplo, abra el símbolo ScrollTrack.

6. Personalice el símbolo como desee.

   Por ejemplo, dibuje un rectángulo negro en medio de la guía; utilice para ello un rectángulo 1 x 4 en las coordenadas (8,0).
7. Repita los pasos 5 y 6 en todos los símbolos que desee personalizar.

   Por ejemplo, dibuje la misma línea en el símbolo ScrollTrackDisabled.
8. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
9. Cree en el escenario una instancia de TextField para introducir datos.
10. Arrastre un componente UIScrollBar a la instancia de TextField.
11. Seleccione Control > Probar película.

Clase UIScrollBar

Herencia MovieClip > Clase UIObject > Clase UIComponent > ScrollBar > UIScrollBar

Nombre de clase de ActionScript mx.controls.UIScrollBar

Las propiedades de la clase UIScrollBar permiten ajustar la posición y la cantidad de desplazamiento cuando el usuario hace clic en las flechas o en la guía de desplazamiento.

A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.controls.UIScrollBar.version);
```

NOTA	El código `trace(myUIScrollBarInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase UIScrollBar

En la tabla siguiente se muestra el método de la clase UIScrollBar.

Método	Descripción
`UIScrollBar.setScrollProperties()`	Establece el intervalo de la barra de desplazamiento y el tamaño del campo de texto asociado a la barra de desplazamiento.
`UIScrollBar.setScrollTarget()`	Asigna la barra de desplazamiento a un campo de texto.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase UIScrollBar de la clase UIObject. Al llamar a estos métodos desde el objeto UIScrollBar, debe utilizarse la forma *UIScrollBarInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase UIScrollBar de la clase UIComponent. Al llamar a estos métodos desde el objeto UIScrollBar, debe utilizarse la forma *UIScrollBarInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase UIScrollBar

En la tabla siguiente se enumeran las propiedades de la clase UIScrollBar.

Propiedad	Descripción
`UIScrollBar.lineScrollSize`	Número de líneas o píxeles de desplazamiento cuando el usuario hace clic en los botones de flecha de la barra de desplazamiento.
`UIScrollBar.pageScrollSize`	Número de líneas o píxeles de desplazamiento cuando el usuario hace clic en la guía de la barra de desplazamiento.
`UIScrollBar.scrollPosition`	Posición de desplazamiento actual de la barra de desplazamiento.
`UIScrollBar._targetInstanceName`	Nombre de instancia del campo de texto asociado a la instancia de UIScrollBar.
`UIScrollBar.horizontal`	Valor booleano que indica si la barra de desplazamiento está orientada verticalmente (`false`), que es la opción predeterminada, u horizontalmente (`true`).

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase UIScrollBar de la clase UIObject. Al acceder a estas propiedades desde el objeto UIScrollBar, debe utilizarse la forma *`UIScrollBarInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.

Propiedad	Descripción
UIObject.scaleY	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
UIObject.top	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
UIObject.visible	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).
UIObject.width	Sólo lectura; anchura del objeto, expresada en píxeles.
UIObject.x	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.y	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase UIScrollBar de la clase UIComponent. Al acceder a estas propiedades desde el objeto UIScrollBar, debe utilizarse la forma *`UIScrollBarInstance.propertyName`*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase UIScrollBar

En la tabla siguiente se muestra el evento de la clase UIScrollBar.

Evento	Descripción
UIScrollBar.scroll	Se difunde al hacer clic en cualquier parte de la barra de desplazamiento.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase UIScrollBar de la clase UIObject.

Evento	Descripción
`UIObject.draw`	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
`UIObject.hide`	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.
`UIObject.load`	Se difunde cuando se crean subobjetos.
`UIObject.move`	Se difunde cuando se mueve el objeto.
`UIObject.resize`	Se difunde cuando cambia el tamaño de un objeto.
`UIObject.reveal`	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
`UIObject.unload`	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase UIScrollBar de la clase UIComponent.

Evento	Descripción
`UIComponent.focusIn`	Se difunde cuando se selecciona un objeto.
`UIComponent.focusOut`	Se difunde cuando un objeto deja de seleccionarse.
`UIComponent.keyDown`	Se difunde cuando se presiona una tecla.
`UIComponent.keyUp`	Se difunde cuando se suelta una tecla.

UIScrollBar.horizontal

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollBarInstance.horizontal

Descripción

Propiedad; indica si la barra de desplazamiento está orientada verticalmente (`false`) u horizontalmente (`true`).

Esta propiedad puede probarse y definirse. El valor predeterminado es `false`.

Ejemplo

En el siguiente ejemplo se utiliza la propiedad `horizontal` para establecer la barra de desplazamiento denominada `my_sb` en una orientación horizontal y se muestra el texto en la instancia `my_txt` del componente TextField:

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/
// Crear el campo de texto.
this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = false;

my_txt.text = "Mary had a little lamb whose fleece " +
"was white as snow and everywhere that Mary went the " +
"lamb was sure to go."

// Crear barra de desplazamiento.
this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);
my_sb.horizontal = true;

// Establecer el campo de texto de destino de la barra de desplazamiento.
my_sb.setScrollTarget(my_txt);
// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(my_txt._width, 16);

// Desplazarlo a la parte inferior del campo de texto.
my_sb.move(my_txt._x, my_txt._y + my_txt._height);
```

UIScrollBar.lineScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollBarInstance.lineScrollSize

Descripción

Propiedad; obtiene o establece el número de líneas o píxeles de desplazamiento cuando el usuario hace clic en los botones de flecha del componente UIScrollBar. Si la barra de desplazamiento está orientada verticalmente, el valor es un número de líneas. Si la barra de desplazamiento está orientada horizontalmente, el valor es un número de píxeles.

El valor predeterminado es 1.

Ejemplo

En el siguiente ejemplo se crea una barra de desplazamiento para desplazar el texto en un campo de texto que se carga desde una página Web. El ejemplo establece la propiedad `lineScrollSize` para desplazarse dos líneas cada vez que se hace clic en un botón de flecha:

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);
```

```
// Desplazarse 2 líneas por cada clic en flecha de desplazamiento.
my_sb.lineScrollSize = 2;

// Desplazarse 5 líneas por cada clic en flecha de desplazamiento.
my_sb.pageScrollSize = 5;

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar.pageScrollSize

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollBarInstance.pageScrollSize

Descripción

Propiedad; obtiene o establece el número de líneas o píxeles de desplazamiento cuando el usuario hace clic en la guía de desplazamiento del componente UIScrollBar. Si la barra de desplazamiento está orientada verticalmente, el valor es un número de líneas. Si la barra de desplazamiento está orientada horizontalmente, el valor es un número de píxeles.

También es posible definir este valor pasando un parámetro *pageSize* con el método `UIScrollBar.setScrollTarget()`.

Ejemplo

En el siguiente ejemplo se crea una barra de desplazamiento para desplazar el texto en un campo de texto que se carga desde una página Web. El ejemplo establece la propiedad `pageScrollSize` para desplazarse cinco líneas de texto cada vez que el usuario hace clic en la guía de desplazamiento:

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Desplazarse 2 líneas por cada clic en flecha de desplazamiento.
my_sb.lineScrollSize = 2;

// Desplazarse 5 líneas por cada clic en guía de desplazamiento.
my_sb.pageScrollSize = 5;

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar.scroll

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
scrollBarInstance.addEventListener("scroll", listenerObject)
```

Sintaxis 2:

```
on (scroll) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) sobre la barra de desplazamiento. La propiedad `UIScrollBar.scrollPosition` y la imagen en pantalla de la barra de desplazamiento se actualizan antes de que se difunda este evento.

El primer ejemplo de sintaxis utiliza un modelo de evento distribuidor/detector en el que el script se sitúa en un fotograma de la línea de tiempo que contiene la instancia del componente. Una instancia de componente (*`scrollBarInstance`*) distribuye un evento (en este caso, `scroll`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se produce el evento. Cuando se produce el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama a `addEventListener()` (véase `EventDispatcher.addEventListener()`) en la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Además de las propiedades normales del objeto de evento (`type` y `target`), el objeto del evento `scroll` incluye una tercera propiedad denominada `direction`. La propiedad `direction` contiene una cadena que describe la orientación de la barra de desplazamiento. Los valores posibles de la propiedad `direction` son `vertical` (valor predeterminado) y `horizontal`.

Para más información sobre las propiedades `type` y `target` del objeto de evento, consulte "Objetos de evento" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente UIScrollBar. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myUIScrollBarComponent` de UIScrollBar, envía "_level0.myUIScrollBarComponent" al panel Salida:

```
on (scroll) {
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se implementa la sintaxis 1 y se crea un objeto detector denominado `sbListener` con un controlador de eventos `scroll`:

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Crear un objeto detector.
var sbListener:Object = new Object();
sbListener.scroll = function(evt_obj:Object){
   // Insertar código para controlar el evento "scroll".
   trace("text is scrolling");
}
```

```
// Añadir detector.
my_sb.addEventListener("scroll", sbListener);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

El código siguiente implementa la sintaxis 2. El código se asocia con la instancia del componente UIScrollBar y envía un mensaje al panel Salida cuando el usuario hace clic en la barra de desplazamiento. El controlador `on()` debe asociarse directamente con la instancia de UIScrollBar.

```
on (scroll) {
  trace("UIScrollBar component was clicked");
}
```

UIScrollBar.scrollPosition

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollBarInstance``.scrollPosition`

Descripción

Propiedad; obtiene o establece la posición de desplazamiento actual del cuadro de desplazamiento (deslizador) cuando se establece un nuevo valor de `scrollPosition`. El valor de `scrollPosition` depende de que la instancia de UIScrollBar se utilice para el desplazamiento vertical u horizontal.

Establezca el desplazamiento de la instancia de destino de la barra de desplazamiento por separado, mediante la siguiente sintaxis:

```
my_scrollbar._targetInstanceName.scroll = 20;
```

Si la instancia de UIScrollBar se utiliza para el desplazamiento vertical (el uso más común), el valor de `scrollPosition` es un entero con un rango entre 0 y el número total de líneas en el campo de texto dividido por el número de líneas que pueden mostrarse simultáneamente en el campo de texto. Si se define un valor de `scrollPosition` en un número por encima de este rango, el campo de texto simplemente se desplazará al final del texto.

Para establecer el cuadro de desplazamiento (deslizador) en la primera línea, defina el valor 0 en `scrollPosition`.

Para establecer el cuadro de desplazamiento (deslizador) en el final, establezca `scrollPosition` en el número de líneas de texto del campo de texto menos uno. Es posible determinar el número de líneas a partir del valor de la propiedad `maxscroll` del campo de texto.

Si la instancia de UIScrollBar se utiliza para el desplazamiento horizontal, el valor de `scrollPosition` es un entero entre 0 y la anchura del campo de texto, en píxeles. Es posible determinar la anchura del campo de texto en píxeles a partir del valor de la propiedad `maxhscroll` del campo de texto.

El valor predeterminado de `scrollPosition` es 0.

Ejemplo

En el ejemplo siguiente, se establece el texto en la posición 20:

```
/**
 Se requiere:
  - Componentes UIScrollBar y Button en la biblioteca
*/
this.createTextField("my_txt", 10, 10, 20, 200, 100);
this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);
this.createClassObject(mx.controls.Button, "my_bt", 30, {label: "Scroll"});

my_txt.wordWrap = true;
my_bt.move(300, 100);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
```

```
my_lv.onData = function(src:String) {
if (src != undefined) {
  my_txt.text = src;
  } else {
    my_txt.text = "Error loading text.";

  }
};

my_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var scroll_listener = new Object();
scroll_listener.click = function() {
  my_sb.scrollPosition = 20;
  my_txt.scroll = 20;
};
my_bt.addEventListener("click", scroll_listener);
```

UIScrollBar.setScrollProperties()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollBarInstance.setScrollProperties(*pageSize*, *minPos*, *maxPos*)

Parámetros

pageSize Número de elementos que pueden verse en el área de visualización. Este parámetro define el tamaño del recuadro de delimitación del campo de texto. Si la barra de desplazamiento es vertical, este valor es un número de líneas de texto; si la barra de desplazamiento es horizontal, este valor es un número de píxeles.

minPos Este parámetro se refiere a la línea de texto de numeración inferior cuando se utiliza la barra de desplazamiento verticalmente o al píxel de numeración inferior del recuadro de delimitación del campo de texto cuando se utiliza la barra de desplazamiento horizontalmente. El valor suele ser 0.

maxPos Este valor se refiere a la línea de texto de numeración superior cuando se utiliza la barra de desplazamiento verticalmente o al píxel de numeración superior del recuadro de delimitación del campo de texto cuando se utiliza la barra de desplazamiento horizontalmente.

Descripción

Método; establece el intervalo de la barra de desplazamiento y el tamaño del campo de texto asociado a la barra de desplazamiento. Este método es útil sobre todo cuando se asocia un componente UIScrollBar con un campo de texto en tiempo de ejecución (mediante `UIScrollBar.setScrollTarget()`) y no durante la edición, y la asignación no provoca que el campo de texto difunda eventos change. Si se utiliza el método `replaceText` para definir el texto del campo de texto, debe utilizarse `setScrollProperties()` para actualizar las barras de desplazamiento.

El componente UIScrollBar utiliza conjuntamente los valores de `minPos` y `maxPos` para determinar el intervalo de desplazamiento de la barra y el campo de texto asociado.

Ejemplo

En el siguiente ejemplo se establece que un componente UIScrollBar muestre 10 líneas de texto simultáneamente en el campo de texto, fuera del rango de 0 a 99 líneas:

```
my_sb.setScrollProperties(10, 0, 99);
```

UIScrollBar.setScrollTarget()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

`scrollBarInstance``.setScrollTarget(`*`textInstance`*`)`

Parámetros

`textInstance` Campo de texto que se asigna a la barra de desplazamiento.

Descripción

Método; asigna un componente UIScrollBar a una instancia de campo de texto. Utilice este método si es necesario asociar un campo de texto y un componente UIScrollBar en tiempo de ejecución.

Ejemplo

En el siguiente ejemplo se crea una barra de desplazamiento para desplazar el texto en un campo de texto que se carga desde una página Web. En el ejemplo se llama al método `setScrollTarget()` para asociar la barra de desplazamiento `my_sb` con el campo de texto `my_txt`.

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Desplazarse 2 líneas por cada clic en flecha de desplazamiento.
my_sb.lineScrollSize = 2;

// Desplazarse 5 líneas por cada clic en guía de desplazamiento.
my_sb.pageScrollSize = 5;

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar._targetInstanceName

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

scrollBarInstance._targetInstanceName

Descripción

Propiedad; indica el nombre de instancia del campo de texto asociado con un componente UIScrollBar. Esta propiedad puede probarse y definirse. Sin embargo, no debería utilizarse para crear una asociación entre un campo de texto y una barra de desplazamiento. Utilice `UIScrollBar.setScrollTarget()` en su lugar.

Ejemplo

En el siguiente ejemplo se crea una barra de desplazamiento para desplazar el texto en un campo de texto que se carga desde una página Web. En el ejemplo se llama a la función `trace()` para mostrar el valor de la propiedad `targetInstanceName`.

```
/**
 Se requiere:
  - Componente UIScrollBar en la biblioteca
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Definir el campo de texto de destino.
my_sb.setScrollTarget(my_txt);

trace(my_sb._targetInstanceName);

// Cambiar el tamaño para ajustar el campo de texto.
my_sb.setSize(16, my_txt._height);

// Desplazarlo junto al campo de texto.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Definir propiedades de desplazamiento.
my_sb.setScrollProperties(10, 0, 99);

// Cargar el texto que debe mostrarse y definir controlador onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
  my_txt.condenseWhite = true;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

CAPÍTULO 56

Clases de servicios Web (sólo en Flash Professional)

56

Las clases de servicios Web, que se encuentran en el paquete mx.services, permiten acceder a servicios Web que utilizan el protocolo SOAP (Simple Object Access Protocol). Esta API no es la misma que la API del componente WebServiceConnector. La API de servicio Web es un conjunto de clases que se pueden utilizar sólo en el código ActionScript y suele incluirse en varios productos Macromedia. Por el contrario, el componente WebServiceConnector es una API exclusiva de Flash que proporciona una interfaz de ActionScript para el componente WebServiceConnector visual.

En la tabla siguiente se enumeran las clases del paquete mx.services. Estas clases funcionan de modo muy integrado, de modo que para aprender sobre las clases de este paquete, debe leer la información en el orden en el que aparece en la tabla.

Clase	Descripción
Clase WebService (sólo en Flash Professional)	Con un archivo del lenguaje de descripción de servicios Web (WSDL, Web Service Definition Language) que define el servicio Web, crea un objeto WebService para llamar a los métodos de servicios Web y para controlar las funciones callback del servicio Web.
Clase PendingCall (sólo en Flash Professional)	Objeto devuelto de una llamada a método de servicio Web que se ha implementado para controlar los resultados y los fallos de la llamada.
Clase Log (sólo en Flash Professional)	Objeto opcional que sirve para registrar la actividad relacionada con un objeto WebService.
Clase SOAPCall (sólo en Flash Professional)	Clase avanzada que contiene información sobre el funcionamiento del servicio Web y que permite controlar determinados comportamientos.

Disponibilidad de clases de servicios Web en tiempo de ejecución (sólo en Flash Professional)

Para que las clases de servicios Web estén disponibles en tiempo de ejecución, el componente WebServiceConnector debe encontrarse en la biblioteca del archivo FLA. Este componente contiene las clases de tiempo de ejecución que permiten trabajar con servicios Web. Para obtener detalles sobre cómo añadir estas clases al archivo FLA, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

NOTA	Estas clases están disponibles automáticamente en un documento de Flash cuando se añade el componente WebServiceConnector al archivo FLA.

Clase Log (sólo en Flash Professional)

Nombre de clase de ActionScript mx.services.Log

La clase Log forma parte del paquete mx.services y se utiliza con la clase WebService (véase "Clase WebService (sólo en Flash Professional)" en la página 1479). Para ver información general sobre las clases del paquete mx.data.services, consulte "Clases de servicios Web (sólo en Flash Professional)" en la página 1455.

Puede crear un objeto Log para registrar la actividad relacionada con un objeto WebService. Para ejecutar código mientras se envían mensajes al objeto Log, utilice la función callback `Log.onLog()`. No hay archivo de registro; el mecanismo de registro es el que haya utilizado durante la función callback `onLog()`, como por ejemplo el envío de mensajes de registro a una sentencia `trace()`.

El constructor de esta clase crea un objeto Log que puede pasarse como un parámetro opcional al constructor de WebService (véase "Clase WebService (sólo en Flash Professional)" en la página 1479).

Resumen de métodos de la clase Log

En la tabla siguiente se enumeran los métodos de la clase PendingCall.

Método	Descripción
`Log.getDateString()`	Devuelve la fecha y hora actual como una cadena con el formato: `mm/dd hh:mm:ss` que se utiliza en los mensajes de registro.
`Log.logInfo()`	Genera un evento `Log.onLog` con un nivel de registro y un mensaje designados.
`Log.logDebug()`	Genera un evento `Log.onLog` con un nivel de registro `Log.DEBUG` y un mensaje designado.

Resumen de propiedades del objeto Log

En la tabla siguiente se enumeran las propiedades de la clase PendingCall.

Propiedad	Descripción
`Log.level`	Categoría de información que desea registrar.
`Log.name`	Nombre de cadena que identifica el objeto Log y que se incluye en cada mensaje del evento `Log.onLog`.

Resumen de funciones callback del objeto Log

En la tabla siguiente se enumeran las funciones callback del objeto Log.

Función callback	Descripción
`Log.onLog()`	Flash Player la llama cuando se envía un mensaje de registro a un archivo de registro.

Constructor de la clase Log

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myWebSrvcLog = new Log([logLevel] [, logName]);
```

Parámetros

logLevel Nivel que indica el tipo de información que desea registrar en el registro. Hay cuatro niveles de registro posibles:

- `Log.BRIEF` Registra el evento de ciclo de vida y las notificaciones de error principales. Éste es el valor predeterminado.
- `Log.VERBOSE` Registra todos los eventos de ciclo de vida y las notificaciones de error.
- `Log.DEBUG` Registra las métricas y los eventos y errores pormenorizados.
- `Log.NONE` No registra nada. Se puede utilizar para desactivar temporalmente los eventos `Log.onLog`.

logName Nombre opcional que se incluye con cada mensaje de registro. Si utiliza varios objetos Log, puede utilizar el nombre de registro para determinar en qué registro ha quedado almacenado un mensaje determinado.

Valor devuelto

Ninguno.

Descripción

Constructor; crea un objeto Log. Una vez que haya creado el objeto Log, puede pasarlo a un servicio Web para recibir los mensajes.

Ejemplo

Puede llamar al constructor `new Log` para que devuelva un objeto Log y lo pase al servicio Web:

```
// Crea un objeto Log.
import mx.services.*;
myWebSrvcLog = new Log();
myWebSrvcLog.onLog = function(msg : String) : Void
{
  myTrace(txt)
}
```

A continuación este objeto Log se pasa en forma de parámetro al constructor de WebService:

```
myWebSrvc = new WebService("http://www.myco.com/info.wsdl", myWebSrvcLog);
```

Cuando se ejecuta el código de los servicios Web y se envían los mensajes al objeto Log, se realiza la llamada a la función `onLog()` de dicho objeto. Aquí es donde debe colocar el código que muestra los mensajes de registro si quiere visualizarlos en tiempo real.

A continuación, se muestran ejemplos de mensajes de registro:

```
7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:44 [INFO] SOAP: Decoded SOAP response into result [16 millis]
7/30 15:22:46 [INFO] SOAP: Received SOAP response from network [6469 millis]
7/30 15:22:46 [INFO] SOAP: Parsed SOAP response XML [15 millis]
7/30 15:22:46 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:46 [INFO] SOAP: Decoded SOAP response into result [16 millis]
```

Log.getDateString()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myWebSrvcLog`.getDateString()`

Parámetros

Ninguno.

Valor devuelto

Fecha y hora actual como una cadena con el formato: `mm/dd hh:mm:ss`.

Descripción

Función; devuelve la fecha y hora actual como una cadena con el formato: `mm/dd hh:mm:ss`. Puede utilizar `Log.getDateString()` para obtener la fecha en el mismo formato que en el mensaje de registro o registrar únicamente la cadena de fecha en un controlador de eventos `log.onLog` para utilizarla en la gestión de registros personalizados.

Ejemplo

En el ejemplo siguiente se crea un objeto Log, se pasa a un objeto WebService nuevo y se controlan los mensajes de registro mediante `Log.getDateString()`, para obtener la hora del registro.

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log(Log.BRIEF, "myLog");
// Pasa el objeto Log al servicio Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
// Controla los mensajes de registro de entrada.
myWebSrvcLog.onLog = function(message : String) : Void
{
  trace("A Log Event Occurred At This Time: "+ this.getDateString());
}
```

Log.logInfo()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myWebSrvcLog`.logInfo(myMessageString)`

Parámetros

msg Mensaje de tipo String que desea que aparezca en el mensaje de evento de registro resultante.

level Nivel que indica el tipo de información que desea registrar en el registro. Hay cuatro niveles de registro posibles:

- `Log.BRIEF` Registra el evento de ciclo de vida y las notificaciones de error principales. Éste es el valor predeterminado.
- `Log.VERBOSE` Registra todos los eventos de ciclo de vida y las notificaciones de error.
- `Log.DEBUG` Registra las métricas y los eventos y errores pormenorizados.
- `Log.NONE` No registra nada. Se puede utilizar para desactivar temporalmente los eventos `Log.onLog`.

Valor devuelto

Ninguno.

Descripción

Función; genera un mensaje de registro establecido por el parámetro `msg`, en un nivel de registro establecido por el parámetro `level`. Este método proporciona una forma de crear sus propios eventos de registro sin ningún nivel de registro.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Log. Se genera un evento `onLog` con un mensaje que indica el inicio de un nuevo registro, mediante una llamada a `Log.logDebug()`.

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log(Log.VERBOSE, "myLog");

// Controla los mensajes de registro de entrada.
myWebSrvcLog.onLog = function(message : String) : Void
{
  trace(message);
}
myWebSrvcLog.logInfo("New Log Started");
// Pasa el objeto Log al servicio Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
```

Log.logDebug()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myWebSrvcLog.logDebug(msg)

Parámetros

msg Cadena de mensaje de registro. La cadena que proporcione en este parámetro aparecerá como mensaje de registro en el evento de registro resultante.

Valor devuelto

Ninguno.

Descripción

Función; genera un mensaje de registro que contiene `msg` y el indicador de tipo de mensaje `[debug]`. Este método proporciona una forma de crear sus propios eventos de registro con `[debug]` en el mensaje de registro, que sólo será visible con un valor de nivel de registro `Log.DEBUG`.

La siguiente cadena es un ejemplo de mensaje de registro de nivel debug, generado por `Log.logDebug()`:

```
12/18 23:20:17 [DEBUG] myLog: My log message
```

Ejemplo

El ejemplo siguiente crea un nuevo objeto Log. Se genera un evento `onLog` con un mensaje que indica el inicio de un nuevo registro, mediante una llamada a `Log.logDebug()`.

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log(Log.DEBUG, "myLog");

// Controla los mensajes de registro de entrada.
myWebSrvcLog.onLog = function(message : String) : Void
{
  trace(message);
}
// Genera un mensaje de registro con un nivel de registro Log.DEBUG.
myWebSrvcLog.logDebug("New Log Started");
// Pasa el objeto Log al servicio Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
```

Log.level

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`myLevel_Number = myWebSrvcLog``.level`

Descripción

Propiedad; indica la categoría de información que desea registrar. Hay cuatro niveles de registro posibles:

- `Log.BRIEF` Registra el evento de ciclo de vida y las notificaciones de error principales. Éste es el valor predeterminado. La propiedad `Log.level` establecida en `Log.BRIEF` devuelve el número 0.
- `Log.VERBOSE` Registra todos los eventos de ciclo de vida y las notificaciones de error. La propiedad `Log.level` establecida en `Log.VERBOSE` devuelve el número 1.
- `Log.DEBUG` Registra las métricas y los eventos y errores pormenorizados. La propiedad `Log.level` establecida en `Log.DEBUG` devuelve el número 2.
- `Log.NONE` No registra nada. Se puede utilizar para desactivar temporalmente los eventos `Log.onLog`. La propiedad `Log.level` establecida en `Log.NONE` devuelve el número -1.

Aunque puede definir esta propiedad directamente, resulta habitual establecer `Log.level` como un parámetro al crear un nuevo objeto Log. Véase "Clase Log (sólo en Flash Professional)" en la página 1456.

Ejemplo

En el siguiente ejemplo se crea un nuevo objeto Log una propiedad `Log.level` establecida en `Log.DEBUG`. Se realiza un seguimiento de la propiedad `Log.level` actual. A continuación, la propiedad `Log.level` del objeto Log se establece en `Log.VERBOSE`.

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log(Log.DEBUG, "myLog");
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);

// Hay que cambiar el nivel del objeto Log.
myWebSrvcLog.level = Log.VERBOSE;
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);
```

Log.name

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myWebServiceName = *myWebSrvcLog*.name

Descripción

Propiedad; cadena que identifica la instancia de Log y que se incluye en cada mensaje del evento `Log.onLog`. Esta propiedad puede obtenerse y definirse. Normalmente se define al crear un nuevo objeto Log. Véase "Clase Log (sólo en Flash Professional)" en la página 1456.

Ejemplo

En el siguiente ejemplo se crea un nuevo objeto Log con la propiedad `Log.level` establecida en `Log.VERBOSE` y con el nombre "myLog". Se realiza un seguimiento de la propiedad `Log.name` actual. A continuación, la propiedad `Log.name` del objeto Log se establece en `"myNewLogName"`.

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log(Log.VERBOSE, "myLog");
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);

// Establece un nuevo nombre para el objeto Log.
myWebSrvcLog.name = "myNewLogName";
trace("myWebSrvcLog.name: " + myWebSrvcLog.name);
```

Log.onLog()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myWebSrvcLog`.onLog = function(message)`

Parámetros

`message` Mensaje de registro que se pasa al controlador. Por ejemplo:

`"7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response"`

Valor devuelto

Ninguno.

Descripción

Función callback; Flash Player la llama cuando se envía un mensaje de registro a un archivo de registro. Esta función es un buen lugar donde colocar el código que registra o muestra los mensajes de registro como, por ejemplo, un comando `trace`. (Para más información sobre la estructura del registro, consulte "Clase Log (sólo en Flash Professional)" en la página 1456.)

Ejemplo

En el ejemplo siguiente se crea un objeto Log, se pasa a un objeto WebService nuevo y se controlan los mensajes de registro:

```
import mx.services.*;
// Crea un objeto Log.
myWebSrvcLog = new Log();
// Pasa el objeto Log al servicio Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
// Controla los mensajes de registro de entrada.
myWebSrvcLog.onLog = function(message : String) : Void
{
  mytrace("myWebSrvcLog.message: " + message);
}
```

Clase PendingCall (sólo en Flash Professional)

Nombre de clase de ActionScript mx.services.PendingCall

La clase PendingCall forma parte del paquete mx.services y se utiliza con la clase WebService. Para ver información general de las clases del paquete mx.services, consulte "Clases de servicios Web (sólo en Flash Professional)" en la página 1455.

No se crea un objeto PendingCall ni se utiliza una función constructora, sino que al llamar a un método en un objeto WebService, el método WebService devuelve un objeto PendingCall. Utilice las funciones callback `PendingCall.onResult` y `PendingCall.onFault` para controlar la respuesta asíncrona del método de servicio Web. Si el método de servicio Web devuelve un fallo, Flash Player llama a `PendingCall.onFault` y pasa un objeto SOAPFault que representa el fallo del protocolo SOAP de XML devuelto por el servidor o servicio Web. El objeto SOAPFault no lo crean directamente los desarrolladores, sino que se devuelve como resultado de un error. Este objeto es una asignación de ActionScript del tipo XML de SOAPFault.

Si la invocación del servicio Web se finaliza correctamente, Flash Player llama a `PendingCall.onResult` y pasa un objeto de resultado. El objeto de resultado es la respuesta XML del servicio Web descodificada y deserializada en ActionScript. Para más información sobre el objeto WebService, consulte "Clase WebService (sólo en Flash Professional)" en la página 1479.

El objeto PendingCall también proporciona acceso a varios parámetros de salida cuando el método de servicio Web devuelve más de un resultado. El valor devuelto al que se hace referencia en esta API es simplemente el primer (o único) resultado; para obtener acceso a todos los resultados, pueden utilizarse las funciones "get output". Por ejemplo, si el valor devuelto en el parámetro a la función callback `onResult` no es el único resultado al que desea tener acceso, puede utilizar `getOutputValues()` (que devuelve una matriz) o `getOutputValue()` (que devuelve un valor individual) para obtener los valores decodificados de ActionScript.

También puede acceder directamente al objeto SOAPParameter. El objeto SOAPParameter es un objeto de ActionScript con dos propiedades: `value` (el valor de ActionScript del parámetro de salida) y `element` (el valor de XML del parámetro de salida). Las siguientes funciones devuelven un objeto SOAPParameter o una matriz de objetos SOAPParameter: `getOutputParameters()`, `getOutputParameterByName()` y `getOutputParameter()`.

Resumen de métodos de la clase PendingCall

En la tabla siguiente se enumeran los métodos de la clase PendingCall.

Método	Descripción
`PendingCall.getOutputParameter()`	Recupera un objeto SOAPParameter por el índice.
`PendingCall.getOutputParameterByName()`	Recupera un objeto SOAPParameter por el nombre.
`PendingCall.getOutputParameters()`	Recupera una matriz de objetos SOAPParameter.
`PendingCall.getOutputValue()`	Recupera el valor de salida correspondiente al índice especificado.
`PendingCall.getOutputValues()`	Recupera una matriz de todos los valores de salida.

Resumen de propiedades del objeto PendingCall

En la tabla siguiente se enumeran las propiedades de la clase PendingCall.

Propiedad	Descripción
`PendingCall.myCall`	Descriptor de operación SOAPCall de la operación PendingCall.
`PendingCall.request`	Petición SOAP en formato XML sin procesar.
`PendingCall.response`	Respuesta SOAP en formato XML sin procesar.

Resumen de funciones callback del objeto PendingCall

En la tabla siguiente se enumeran las funciones callback de la clase PendingCall.

Función callback	Descripción
`PendingCall.onFault`	Flash Player la llama cuando un método de servicio Web ha fallado y ha devuelto un error.
`PendingCall.onResult`	Se realiza esta llamada cuando un método se ha llevado a cabo correctamente y ha devuelto un resultado.

PendingCall.getOutputParameter()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myPendingCall.getOutputParameter(*index*)

Parámetros

index Índice basado en cero del parámetro.

Valor devuelto

Un objeto SOAPParameter con dos propiedades: `value` (el valor de ActionScript del parámetro de salida) y `element` (el valor de XML del parámetro de salida).

Descripción

Función; obtiene un parámetro de salida adicional del objeto SOAPParameter, el cual contiene el valor y el elemento XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto se proporciona siempre en el parámetro *result* de la función callback `onResult`, pero para obtener acceso a los otros valores devueltos deben utilizarse funciones como `getOutputParameter()` y `getOutputValue()`. La función `getOutputParameter()` devuelve el parámetro de salida *n* en forma de objeto SOAPParameter.

Ejemplo

Un descriptor SOAP como el siguiente, `getOutputParameter(1)` devolverá un objeto SOAPParameter con `value="Hi there!"` y `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Véase también

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputParameterByName()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myPendingCall.getOutputParameterByName(var *localName*)

Parámetros

localName Nombre local del parámetro. En otras palabras, el nombre de un elemento XML, sin información del espacio de nombres. Por ejemplo, el nombre local de los dos elementos siguientes es `bob`:

```
<bob abc="123">
<xsd:bob def="ghi">
```

Valor devuelto

Un objeto SOAPParameter con dos propiedades: `value` (el valor de ActionScript del parámetro de salida) y `element` (el valor de XML del parámetro de salida).

Descripción

Función; obtiene los parámetros de salida en forma de objeto SOAPParameter, que contiene el valor y el elemento XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto se proporciona siempre en el parámetro *result* de la función callback `onResult`, pero para obtener acceso a los otros valores devueltos deben utilizarse funciones como `getOutputParameterByName()`. Esta función devuelve el parámetro de salida con el nombre *localName*.

Ejemplo

Un descriptor SOAP como el siguiente, `getOutputParameterByName("outParam2")` devolverá un objeto SOAPParameter con `value="Hi there!"` y `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Véase también

`PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputParameters()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myPendingCall.getOutputParameters()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto SOAPParameter con dos propiedades: `value` (el valor de ActionScript del parámetro de salida) y `element` (el valor de XML del parámetro de salida).

Descripción

Función; obtiene parámetros de salida adicionales del objeto SOAPParameter, el cual contiene los valores y los elementos XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto se proporciona siempre en el parámetro *`result`* de la función callback `onResult`, pero para obtener acceso a los otros valores devueltos deben utilizarse funciones como `getOutputParameters()` y `getOutputValues()`.

Véase también

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputValue()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myPendingCall.getOutputValue(var index)
```

Parámetros

index Índice de un parámetro de salida. El primer parámetro es el índice 0.

Valor devuelto

Parámetro de salida *n*.

Descripción

Función; obtiene el valor de ActionScript descodificado de un parámetro de salida individual. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto se proporciona siempre en el parámetro *result* de la función callback `onResult`, pero para obtener acceso a los otros valores devueltos deben utilizarse funciones como `getOutputValue()` y `getOutputParameter()`. La función `getOutputValue()` devuelve el parámetro de salida *n*.

Ejemplo

Un archivo SOAP como el siguiente, `getOutputValue(2)`, devolverá `true`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Véase también

`PendingCall.getOutputParameterByName()`,`PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValues()`

PendingCall.getOutputValues()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

myPendingCall.getOutputValues()

Parámetros

Ninguno.

Valor devuelto

Matriz de los valores descodificados de todos los parámetros de salida.

Descripción

Función; obtiene el valor de ActionScript descodificado de todos los parámetros de salida. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto se proporciona siempre en el parámetro *result* de la función callback `onResult`, pero para obtener acceso a los otros valores devueltos deben utilizarse funciones como `getOutputValues()` y `getOutputParameters()`.

Véase también

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`

PendingCall.myCall

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

PendingCall.myCall

Descripción

Propiedad; objeto SOAPCall correspondiente a la operación PendingCall. El objeto SOAPCall contiene información sobre la operación del servicio Web y permite controlar determinados comportamientos. Para más información, consulte "Clase SOAPCall (sólo en Flash Professional)" en la página 1476.

Ejemplo

La función callback `onResult` siguiente rastrea el nombre de la operación SOAPCall.

```
callback.onResult = function(result)
{
  // Comprobar el nombre de la operación.
  trace("My operation name is " + this.myCall.name);
}
```

PendingCall.onFault

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myPendingCallObj.onFault = function(fault)
{
  // El código se escribe aquí.
}
```

Parámetros

fault Versión del objeto de ActionScript descodificado del objeto SOAPFault con propiedades. Si la información sobre el error proviene de un servidor con el formato XML, el objeto SOAPFault será la versión de ActionScript descodificada de dicho XML.

El tipo de objeto de error devuelto a `PendingCall.onFault` es un objeto SOAPFault. No lo crea directamente el desarrollador, sino que se devuelve como resultado de un error. Este objeto es una asignación de ActionScript del tipo XML de SOAPFault.

Propiedad SOAPFault	Descripción
`faultcode`	Cadena; cadena breve que describe el error.
`faultstring`	Cadena; descripción inteligible del error.
`detail`	Cadena; información específica de la aplicación asociada con el error, como por ejemplo un seguimiento de la pila u otra información que devuelve el motor del servicio Web.
`element`	XML; objeto XML que representa la versión XML del fallo.
`faultactor`	Cadena; origen del fallo (opcional si no interviene ningún intermediario).

Valor devuelto

Ninguno.

Descripción

Función callback; Flash Player la llama cuando un método de servicio Web ha fallado y ha devuelto un error. El parámetro *fault* es un objeto SOAPFault de ActionScript.

Es un buen lugar para insertar código que controle los fallos, por ejemplo, para indicar al usuario que el servidor no está disponible o para contactar con el servicio de asistencia técnica en caso necesario.

Ejemplo

En el ejemplo siguiente se controlan los errores devueltos desde el método de servicio Web.

```
// Controla todos los errores devueltos debido al uso de un método de
  servicio Web.
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onFault = function(fault)
{
  // Detecta el fallo de SOAP.
  DebugOutputField.text = fault.faultstring;

  // Añadir código para controlar los fallos, por ejemplo indicando al
  // usuario que el servidor no está disponible o que contacte
  // con el servicio de asistencia técnica.

}
```

PendingCall.onResult

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myPendingCallObj.onResult = function(result)
{
  // El código se escribe aquí.

}
```

Parámetros

result Versión del objeto de ActionScript descodificado del resultado XML devuelto por un método de servicio Web llamado mediante `myPendingCallObj = myWebService.methodName(params)`.

Valor devuelto

Ninguno.

Descripción

Función callback; Flash Player llama a esta función cuando un método de servicio Web funciona correctamente y devuelve un resultado. El resultado es la versión del objeto de ActionScript descodificado del XML devuelto por la operación. En esta función, debe incluirse código que lleve a cabo la función apropiada en función del resultado. Para devolver XML sin procesar en lugar del resultado descodificado, acceda a la propiedad `PendingCall.response`.

Ejemplo

En el ejemplo siguiente se controlan los resultados devueltos del método de servicio Web.

```
// Controla los resultados devueltos debido al uso de un método de servicio
  Web.
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onResult = function(result)
{
  // Detecta el resultado y lo controla para esta aplicación.
  ResultOutputField.text = result;
}
```

Véase también

`PendingCall.response`

PendingCall.request

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
rawXML = myPendingCallback.request;
```

Descripción

Propiedad; contiene el formato XML sin procesar de la petición actual enviada con `myPendingCallback = myWebService.methodName()`. Normalmente, no se utiliza `PendingCall.request`, pero puede resultar útil si está interesado en las comunicaciones SOAP que se envían por la red. Para obtener la versión de ActionScript de los resultados de la petición, utilice `myPendingCallback.onResult`.

PendingCall.response

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
rawXML = myPendingCallback.response;
```

Descripción

Propiedad; contiene formato XML sin procesar de la respuesta a la llamada al método de servicio Web más reciente enviada con `myPendingCallback = myWebService.methodName()`. Normalmente, no se utiliza `PendingCall.response`, pero puede resultar útil si está interesado en las comunicaciones SOAP que se envían por la red. Para obtener la versión de ActionScript correspondiente de los resultados de la petición, utilice `myPendingCallback.onResult`.

Clase SOAPCall (sólo en Flash Professional)

Nombre de clase de ActionScript mx.services.SOAPCall

La clase SOAPCall forma parte del paquete mx.services y es una clase avanzada que se utiliza con la clase WebService (véase "Clase WebService (sólo en Flash Professional)" en la página 1479). Para ver información general sobre las clases del paquete mx.data.services, consulte "Clases de servicios Web (sólo en Flash Professional)" en la página 1455.

Los desarrolladores no crean objetos SOAPCall. Cuando se llama a un método para un objeto WebService, dicho objeto devuelve un objeto PendingCall. Para acceder al objeto SOAPCall asociado, utilice `myPendingCall.myCall`.

Cuando se crea un objeto WebService, éste contiene los métodos correspondientes a las operaciones de la URL WSDL que se pasen. En segundo plano, también se crea un objeto SOAPCall para cada operación en WSDL. El objeto SOAPCall es el descriptor de la operación y como tal contiene toda la información sobre dicha operación determinada; es decir, el aspecto del XML en la red, el estilo de la operación, etc. También permite controlar determinados comportamientos. Puede obtenerse el objeto SOAPCall de una determinada operación mediante la función `WebService.getCall()`. Existe un solo objeto SOAPCall para cada operación, que comparten todas las llamadas activas a dicha operación. Una vez que disponga del objeto SOAPCall, puede personalizar el descriptor realizando las acciones siguientes:

- Activar/desactivar la descodificación de la respuesta XML
- Activar/desactivar el retardo de la conversión de matrices SOAP en objetos de ActionScript
- Modificar la configuración de simultaneidad de una operación determinada

Resumen de propiedades del objeto SOAPCall

En la siguiente tabla se enumeran las propiedades del objeto SOAPCall.

Propiedad	Descripción
`SOAPCall.concurrency`	Número de peticiones simultaneas.
`SOAPCall.doDecoding`	Activa o desactiva la descodificación de la respuesta XML.
`SOAPCall.doLazyDecoding`	Activa o desactiva la "descodificación diferida" (el retardo de la conversión de matrices SOAP en objetos de ActionScript).

SOAPCall.concurrency

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

SOAPCall.concurrency

Descripción

Propiedad; número de peticiones simultaneas. Los valores posibles aparecen enumerados en la tabla siguiente:

Valor	Descripción
`SOAPCall.MULTIPLE_CONCURRENCY`	Permite varias llamadas activas.
`SOAPCall.SINGLE_CONCURRENCY`	Permite una sola llamada cada vez e indica un error cuando hay una llamada activa.
`SOAPCall.LAST_CONCURRENCY`	Permite una sola llamada mediante la cancelación de las anteriores.

SOAPCall.doDecoding

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`SOAPCall``.doDecoding`

Descripción

Propiedad; activa la descodificación de la respuesta XML (`true`) o la desactiva (`false`). De forma predeterminada, la respuesta XML se convierte (descodifica) en objetos de ActionScript. Si sólo desea el código XML, establezca `SOAPCall.doDecoding` en `false`.

SOAPCall.doLazyDecoding

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`SOAPCall``.doLazyDecoding`

Descripción

Propiedad; activa la "descodificación diferida" de las matrices (`true`) o la desactiva (`false`). De forma predeterminada, se utiliza un algoritmo de "descodificación diferida" para retrasar hasta el último momento la conversión de las matrices SOAP en objetos de ActionScript; de esta forma, las funciones se ejecutan mucho más rápido cuando devuelven grandes juegos de datos. Esto significa que las matrices que se reciben de la ubicación remota son objetos ArrayProxy. Después, cuando se accede a un índice determinado (`foo[5]`) dicho elemento se descodifica automáticamente, si es necesario. Este comportamiento puede desactivarse (lo que provocará la completa descodificación de todas las matrices) estableciendo `SOAPCall.doLazyDecoding` en `false`.

Clase WebService (sólo en Flash Professional)

Nombre de clase de ActionScript mx.services.WebService

La clase WebService forma parte del paquete mx.services y se utiliza con las clases Log, PendingCall y SOAPCall. Para ver información general de las clases del paquete mx.services, consulte "Clases de servicios Web (sólo en Flash Professional)" en la página 1455.

NOTA	Esta clase WebService no es la misma que la clase WebServiceConnector. La clase WebServiceConnector proporciona una interfaz de ActionScript al componente WebServiceConnector visual.

El objeto WebService actúa como una referencia local a un servicio Web remoto. Cuando se crea un objeto WebService, el archivo WSDL que define el servicio Web se descarga, se analiza y se coloca en el objeto. Después, se llaman directamente a los métodos del servicio Web para el objeto WebService y se controlan las funciones callback del servicio Web. Si WSDL se ha procesado correctamente y el objeto WebService está listo, se invoca la función callback `WebService.onLoad`. Si se detecta un problema durante la carga del WSDL, se invoca la función callback `WebService.onFault`.

Cuando se llama a un método para un objeto WebService, el valor devuelto es un objeto de función callback. El tipo de objeto de la función callback devuelto de todas los métodos de servicio Web es `PendingCall`. Estos objetos no suelen crearlos los desarrolladores, sino que se crean automáticamente como resultado del método *`webServiceObject.webServiceMethodName()`* que se ha llamado. Estos objetos no son el resultado de la llamada de WebService, que se produce posteriormente, sino que el objeto PendingCall representa la llamada en curso. Una vez finalizada la ejecución de la operación de WebService (generalmente segundos después de realizar una llamada a un método), se rellenan los diferentes campos de datos de PendingCall y se llama a la función callback `PendingCall.onResult` o `PendingCall.onFault` proporcionada. Para más información sobre el objeto PendingCall, consulte "Clase PendingCall (sólo en Flash Professional)" en la página 1465.

Flash Player pone en cola las llamadas que se hayan realizado antes de haber analizado el WSDL e intenta ejecutarlas después de analizar el WSDL. Esto se debe a que el WSDL contiene información necesaria para que las peticiones SOAP se codifiquen y envíen correctamente. No es necesario poner en cola las llamadas de función que se realicen una vez que se haya analizado el WSDL, pues se ejecutan de forma inmediata. Si una llamada en cola no coincide con el nombre de ninguna de las operaciones definidas en el WSDL, Flash Player devuelve un error al objeto de función callback que se proporcionó al realizar la llamada originalmente.

La interfaz API WebServices, que se incluye en el paquete mx.services, consta de las clases WebService, Log y PendingCall.

Para que las clases de servicios Web estén disponibles en tiempo de ejecución, el componente WebServiceConnector debe encontrarse en la biblioteca del archivo FLA. Si se utiliza ActionScript únicamente para acceder a un servicio Web en tiempo de ejecución, se debe añadir este componente manualmente a la biblioteca del documento. Para más información sobre cómo añadir este componente al documento, consulte Capítulo 16, "Integración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Resumen de métodos del objeto WebService

En la tabla siguiente se enumeran los métodos del objeto WebService.

Método	Descripción
`WebService.getCall()`	Obtiene el objeto SOAPCall de una operación determinada.
`WebService.myMethodName()`	Invoca una operación de servicio Web específica definida por el WSDL.

Resumen de las funciones callback del objeto WebService

En la tabla siguiente se enumeran las funciones callback del objeto WebService.

Función callback	Descripción
`WebService.onFault`	Se llama cuando se produce un error durante el análisis de WSDL.
`WebService.onLoad`	Se llama cuando el servicio Web ha cargado y analizado correctamente su archivo WSDL.

Tipos compatibles (sólo en Flash Professional)

Las clases de servicios Web admiten un subconjunto de tipos de esquema XML (tipos de datos), como se define en las tablas siguientes.

Los tipos de datos complejos y el tipo de matriz codificada con SOAP también son compatibles y pueden estar formados por otros tipos complejos, matrices o tipos de esquema XML incorporados:

- "Tipos numéricos simples" en la página 1481
- "Tipos simples de fecha y hora" en la página 1482
- "Tipos simples de nombre y cadena" en la página 1482
- "Tipo booleano" en la página 1483
- "Tipos de objeto" en la página 1483
- "Elementos compatibles del objeto de esquema XML" en la página 1483

Tipos numéricos simples

Tipo de esquema XML	Vinculación de ActionScript
`decimal`	Number
`integer`	Number
`negativeInteger`	Number
`nonNegativeInteger`	Number
`positiveInteger`	Number
`long`	Number

Tipo de esquema XML	Vinculación de ActionScript
int	Number
short	Number
byte	Number
unsignedLong	Number
unsignedShort	Number
unsignedInt	Number
unsignedByte	Number
float	Number
double	Number

Tipos simples de fecha y hora

Tipo de esquema XML	Vinculación de ActionScript
date	Objeto Date
datetime	Objeto Date
duration	Objeto Date
gDay	Objeto Date
gMonth	Objeto Date
gMonthDay	Objeto Date
gYear	Objeto Date
gYearMonth	Objeto Date
time	Objeto Date

Tipos simples de nombre y cadena

Tipo de esquema XML	Vinculación de ActionScript
cadena	Cadena de ActionScript
normalizedString	Cadena de ActionScript
QName	Objeto mx.services.Qname

Tipo booleano

Tipo de esquema XML	Vinculación de ActionScript
`Boolean`	Boolean

Tipos de objeto

Tipo de esquema XML	Vinculación de ActionScript
`Any`	Objeto XML
`Complex Type`	Objeto de ActionScript compuesto por propiedades de cualquier tipo compatible
`Array`	Matriz de ActionScript compuesta por cualquier objeto o tipo compatible

Elementos compatibles del objeto de esquema XML

En la descripción del siguiente esquema se ilustran los elementos compatibles del objeto de esquema XML:

```
schema
     complexType
          complexContent
               restriction
          sequence | simpleContent
               restriction
     element
          complexType | simpleType
```

Seguridad de WebService (sólo en Flash Professional)

Los métodos y funciones callback de la clase WebService cumplen con el modelo de seguridad de Flash Player. Para más información sobre el modelo de seguridad de Flash Player, consulte "Aspectos básicos de la seguridad" en *Aprendizaje de ActionScript 2.0 en Flash*.

Autenticación y autorización de usuarios Las reglas de autenticación y autorización son las mismas para la interfaz API de WebService que para cualquier operación de red XML de Flash. El propio protocolo SOAP no especifica ningún método de autenticación ni autorización. Por ejemplo, cuando el transporte HTTP subyacente devuelve una respuesta HTTP BASIC en los encabezados HTTP, el navegador responde presentando un cuadro de diálogo al usuario y adjuntando seguidamente la entrada del usuario a los encabezados HTTP en mensajes posteriores. Este mecanismo se produce en un nivel inferior al de SOAP y forma parte del diseño de autenticación HTTP de Flash.

Integridad de los mensajes La seguridad de los mensajes implica el cifrado de los propios mensajes SOAP en una capa conceptual sobre los paquetes de red en los que se entregan los mensajes SOAP.

Seguridad de transporte El transporte de red subyacente para los servicios Web SOAP de Flash Player siempre es HTTP `POST`. Por lo tanto, los medios de seguridad que se pueden aplicar en la capa de transporte HTTP de Flash, como SSL, se admiten a través de invocaciones de servicios Web desde Flash. SSL/HTTPS proporciona el tipo más común de seguridad de transporte para la mensajería SOAP; además, la utilización de autenticación HTTP BASIC, junto con SSL en la capa de transporte, es el tipo de seguridad más común para sitios Web en la actualidad.

Constructor de la clase WebService

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myWebServiceObject = new WebService(wsdlURI [, logObject]);
```

Parámetros

`wsdlURI` URI del archivo WSDL del servicio Web.

`logObject` Parámetro opcional que especifica el nombre del objeto Log de este servicio Web. Si se utiliza este parámetro, debe crearse primero el objeto Log. Para más información, consulte “Clase Log (sólo en Flash Professional)” en la página 1456.

Valor devuelto

Ninguno.

Descripción

Función constructora; crea un objeto WebService. Cuando se llama a `new WebService()`, se proporciona una URL de WSDL y Flash Player devuelve un objeto WebService. El constructor puede aceptar además un URI de proxy y un objeto Log.

Si lo desea, puede utilizar dos funciones callback para el objeto WebService. Flash Player llama a *`webServiceObject`*`.onLoad` cuando acaba de analizar el archivo WSDL y el objeto se ha completado. Es una buena ubicación para incluir código que se desea ejecutar sólo después de que el archivo WSDL se haya analizado completamente. Por ejemplo, puede incluir la primera llamada a método de servicio Web en esta función.

Flash Player llama a *`webServiceObject`*`.onFault` cuando se produce un error al buscar o analizar el archivo WSDL. Es una buena ubicación para incluir código de depuración y código que indique al usuario que el servidor no está disponible, que se vuelva a intentar la acción más tarde o información similar. Para más información, consulte las entradas individuales de estas funciones.

Invocación de una operación de servicio Web Las operaciones de servicio Web se invocan como método directamente disponible en el servicio Web. Por ejemplo, si el servicio Web tiene el método `getCompanyInfo(tickerSymbol)`, se invocaría el método de la siguiente manera:

```
myPendingCallObject = myWebServiceObject.getCompanyInfo(tickerSymbol);
```

En este ejemplo, el objeto de función callback se denomina `myPendingCallObject`. Todas las invocaciones de métodos son asíncronas y devuelven un objeto de función callback del tipo `PendingCall`. (*Asíncrono* significa que el resultado de la llamada al servicio Web no está disponible de forma inmediata.)

Observe la siguiente llamada:

```
x = stockService.getQuote("macr");
```

Cuando se realiza esta llamada, el objeto `x` no es el resultado de `getQuote`, sino que es un objeto PendingCall. Los resultados reales sólo se encuentran disponibles posteriormente, cuando finaliza la operación del servicio Web. El código de ActionScript se notifica mediante una llamada a la función callback `onResult`.

Gestión del objeto PendingCall Este objeto de función callback es un objeto PendingCall que se utiliza para gestionar los resultados y los errores del método de servicio Web llamado (véase "Clase PendingCall (sólo en Flash Professional)" en la página 1465). A continuación, se muestra un ejemplo:

```
MyPendingCallObject = myWebServiceObject.myMethodName(param1, ..., paramN);
MyPendingCallObject.onResult = function(result)
{
  OutputField.text = result
}
MyPendingCallObject.onFault = function(fault)
{
  DebugField.text = fault.faultCode + "," + fault.faultstring;

  // Añadir código para controlar los fallos, por ejemplo indicando al
  // usuario que el servidor no está disponible o que contacte
  // con el servicio de asistencia técnica.

}
```

WebService.getCall()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
getCall(var operationName)
```

Parámetros

operationName Operación de servicio Web del objeto SOAPCall correspondiente que desea recuperar.

Valor devuelto

Un objeto SOAPCall.

Descripción

Función; cuando se crea un objeto WebService, éste contiene los métodos correspondientes a las operaciones de la URL de WSDL que se pasen. En segundo plano, también se crea un objeto SOAPCall para cada operación en el archivo WSDL. El objeto SOAPCall es el descriptor de la operación y como tal contiene toda la información sobre dicha operación determinada; es decir, el aspecto del XML en la red, el estilo de la operación, etc. También permite controlar determinados comportamientos. Puede obtener el objeto SOAPCall de una operación determinada utilizando el método `getCall()`. Existe un solo objeto SOAPCall para cada operación, que comparten todas las llamadas activas a dicha operación. Una vez que dispone del objeto SOAPCall, puede cambiar el descriptor de operador utilizando la clase SOAPCall; véase "Clase SOAPCall (sólo en Flash Professional)" en la página 1476.

WebService.*myMethodName*()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
callbackObj = myWebServiceObject.myMethodName(param1, ... paramN);
```

Parámetros

`param1, ... paramN` Diversos parámetros, en función del método de servicio Web que se llame.

Valor devuelto

Objeto PendingCall con el que se puede asociar una función para gestionar los resultados y los errores de la invocación. Para más información, consulte "Clase PendingCall (sólo en Flash Professional)" en la página 1465.

La función callback que se invoca cuando se devuelve la respuesta del método WebService es `PendingCall.onResult` o `PendingCall.onFault`. Si identifica de forma exclusiva los objetos de función callback, puede gestionar varias funciones callback `onResult`, como en el ejemplo siguiente:

```
myWebService = new WebService("http://www.myCompany.com/myService.wsdl");
callback1 = myWebService.getWeather("02451");
callback1.onResult = function(result)
{
  // realizar una acción
}
callback2 = myWebService.getDetailedWeather("02451");
callback2.onResult = function(result)
{
  // realizar otra acción
}
```

Descripción

Método; invoca una operación de servicio Web. El método se invoca directamente en el servicio Web. Por ejemplo, si el servicio Web tiene el método `getCompanyInfo(tickerSymbol)`, realizaría la siguiente llamada:

```
myCallbackObject.myservice.getCompanyInfo(tickerSymbol);
```

Todas las invocaciones son asíncronas y devuelven un objeto de función callback del tipo `PendingCall`.

WebService.onFault

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

```
Usage
MyWebServiceObject.onFault = function(fault)
{
  // El código se escribe aquí.
}
```

Parámetros

`fault` Versión del objeto de ActionScript descodificado del error con propiedades. Si la información sobre el error proviene de un servidor con el formato XML, el objeto SOAPFault será la versión de ActionScript descodificada de dicho XML.

El tipo de objeto de error que se devuelve a los métodos `WebService.onFault` es un objeto SOAPFault. No lo crean directamente los desarrolladores, sino que se devuelve como resultado de un error. Este objeto es una asignación de ActionScript del tipo XML de SOAPFault.

Propiedad SOAPFault	Descripción
`faultcode`	Cadena; QName estándar más corto que describe el error.
`faultstring`	Cadena; descripción inteligible del error.
`detail`	Cadena; información específica de la aplicación asociada con el error, como por ejemplo un seguimiento de la pila u otra información que devuelve el motor del servicio Web.
`element`	XML; objeto XML que representa la versión XML del fallo.
`faultactor`	Cadena; origen del fallo. Es opcional si no interviene ningún intermediario.

Valor devuelto

Ninguno.

Descripción

Función callback; Flash Player la llama cuando el constructor `new WebService()` ha fallado y ha devuelto un error. Esto puede suceder cuando el archivo WSDL no se puede analizar o no se puede encontrar. El parámetro *`fault`* es un objeto SOAPFault de ActionScript.

Ejemplo

En el ejemplo siguiente se gestionan los errores devueltos en la creación del objeto WebService.

```
MyWebServiceObject.onFault = function(fault)
{
  // Captura el fallo.
  DebugOutputField.text = fault.faultstring;

  // Añadir código para controlar los fallos, por ejemplo indicando al
  // usuario que el servidor no está disponible o que contacte
  // con el servicio de asistencia técnica.

}
```

WebService.onLoad

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
myService.onLoad = function(wsdlDocument)
{
  // El código se escribe aquí.
}
```

Parámetros

wsdlDocument Documento WSDL XML.

Valor devuelto

Ninguno.

Descripción

Función callback; Flash Player la llama cuando el objeto WebService ha cargado y analizado correctamente su archivo WSDL. Si se invocan operaciones en una aplicación antes de llamar a esta función callback, se ponen en cola internamente y no se transiten realmente hasta que se ha cargado el WSDL.

Ejemplo

En el ejemplo siguiente se especifica la URL de WSDL, se crea un objeto de servicio Web y se recibe el documento WSDL después de cargarlo.

```
// Especificar la URL de WSDL.
var wsdlURI = "http://www.flash-db.com/services/ws/companyInfo.wsdl";

// Crea un objeto de servicio Web.
stockService = new WebService(wsdlURI);

// Recibe el documento WSDL después de cargarlo.
stockService.onLoad = function(wsdlDocument);
{
  // Código que debe ejecutarse cuando la carga de WSDL finalice y el
  // objeto se haya creado.
}
```

CAPÍTULO 57

Componente WebServiceConnector (sólo en Flash Professional)

57

El componente WebServiceConnector le permite acceder a métodos remotos presentados por un servidor mediante el protocolo estándar SOAP (Simple Object Access Protocol). Un método de servicio Web puede aceptar parámetros y devolver un resultado. Utilizando la herramienta de edición de Flash Professional 8 y el componente WebServiceConnector, puede inspeccionar, acceder a y vincular datos entre un servidor Web remoto y su aplicación Flash.

Una sola instancia de componente WebServiceConnector puede utilizarse para hacer diversas llamadas a la misma operación. Sin embargo, tiene que utilizar una instancia de WebServiceConnector distinta para cada operación distinta que desee llamar.

NOTA El componente WebServiceConnector aparece en el escenario durante la edición de la aplicación, pero es visible en la aplicación en tiempo de ejecución.

Para más información de introducción sobre el trabajo con los resultados de este componente, consulte "Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)" en *Utilización de Flash*.

Utilización del componente WebServiceConnector (sólo en Flash Professional)

Puede utilizar el componente WebServiceConnector para conectarse a un servicio Web y permitir que sus propiedades puedan vincularse a propiedades de componentes de interfaz de usuario de la aplicación. Para conectarse a un servicio Web, debe introducir primero la URL para el archivo WSDL que representa el servicio Web. Puede introducir esta URL en el inspector de componentes del panel Servicios Web. Consulte "Conexión a servicios Web con el componente connector WebService (sólo para Flash Professional)" en *Utilización de Flash*.

Para más información sobre la conexión a los servicios Web, consulte "Vinculación de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Parámetros de WebServiceConnector

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente WebServiceConnector mediante la ficha Parámetros del inspector de componentes:

multipleSimultaneousAllowed es un valor booleano que indica si se pueden realizar varias llamadas a la vez; el valor predeterminado es false. Si su valor es `false`, el método `trigger()` no realizará una llamada si ya se está realizando otra llamada. Se emitirá un evento de estado, con el código `CallAlreadyInProgress`. Si este parámetro es `true`, la llamada tiene lugar.

operation es una cadena que indica el nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.

suppressInvalidCalls es un valor booleano que indica si debe suprimirse una llamada si los parámetros no son válidos; el valor predeterminado es `false`. Si su valor es `true`, el método `trigger()` no realizará una llamada si los parámetros vinculados a datos no superan la validación Se emitirá un evento de estado, con el código `InvalidParams`. Si este parámetro es `false`, la llamada tiene lugar con los datos no válidos, del modo requerido.

WSDLURL (tipo cadena) es la URL del archivo WSDL que define la operación de servicio Web. Cuando se define esta URL durante la edición, el archivo WSDL se recoge y se analiza de forma inmediata. Los parámetros resultantes y la información de resultados se pueden consultar en la ficha Esquema del inspector de componentes. La descripción del servicio también se añade al panel Servicio Web. Por ejemplo, véase www.flash-mx.com/mm/tips/tips.cfc?wsdl.

Flujo de trabajo normal del componente WebServiceConnector

En el siguiente procedimiento se muestra el flujo de trabajo típico del componente WebServiceConnector.

Para utilizar un componente WebServiceConnector:

1. Utilice el panel Servicios Web para introducir la URL de un archivo WSDL de un servicio Web.
2. Para añadir una llamada a un método del servicio Web, seleccione el método, haga clic con el botón derecho del ratón (Windows) o presione la tecla Control y haga clic (Macintosh) y, a continuación, seleccione Añadir llamada de método del menú contextual.

Esto crea una instancia del componente WebServiceConnector en su aplicación. El esquema para el componente puede encontrarse en la ficha Esquema del inspector de componentes. Puede utilizar este esquema a su gusto; por ejemplo, para realizar cambios adicionales en el formato o en la configuración de validación.

NOTA	El esquema para las propiedades del componente `params` y `results` se actualiza cada vez que el usuario cambia el parámetro `WSDLURL` u `operation`. Esto sobrescribe cualquier cambio que se haya realizado.

3. Utilice la ficha Vinculaciones del inspector de componentes para vincular los parámetros del servicio Web y los resultados que ahora están definidos en su esquema con los componentes de interfaz de usuario de su aplicación.
4. Utilice un desencadenante para iniciar la operación de vinculación de datos de una de las siguientes formas:
   - Asocie el comportamiento Trigger Data Source a un botón.
   - Utilice su propio código ActionScript para llamar al método `trigger()` en el componente WebServiceConnector.
   - Cree una vinculación entre el parámetro de servicio Web y un control de interfaz de usuario y establezca su propiedad Kind en AutoTrigger. Para más información, consulte "Tipos de esquema" en *Utilización de Flash*.

Para ver un ejemplo paso a paso que conecte y muestre un servicio Web mediante el componente WebServiceConnector, consulte el "Web Service Tutorial: Macromedia Tips".

Clase WebServiceConnector (sólo en Flash Professional)

Herencia RPC > WebServiceConnector

Nombre de clase de ActionScript mx.data.components.WebServiceConnector

Esta clase permite conectar a servicios Web remotos mediante código ActionScript en lugar de utilizar instancias de componente en el escenario. Para utilizar la clase WebServiceConnector, es necesario añadir una instancia del componente WebServiceConnector a su biblioteca. No es necesario colocar el componente directamente en el escenario. Hay que importar la clase de ActionScript mx.data.components.WebServiceConnector al principio del script o utilizar el nombre de clase completo en el código.

Resumen de métodos de la clase WebServiceConnector

En la tabla siguiente se muestra el método de la clase WebServiceConnector.

Método	Descripción
`WebServiceConnector.trigger()`	Inicia una llamada a un servicio Web.

Resumen de propiedades de la clase WebServiceConnector

En la tabla siguiente se enumeran las propiedades de la clase WebServiceConnector.

Propiedad	Descripción
`WebServiceConnector.multipleSimultaneousAllowed`	Indica si se pueden realizar varias llamadas a la vez.
`WebServiceConnector.operation`	Indica el nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.
`WebServiceConnector.params`	Especifica los datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`.
`WebServiceConnector.results`	Identifica los datos que se han recibido del servidor como resultado de la operación `trigger()`.
`WebServiceConnector.suppressInvalidCalls`	Indica si debe suprimirse una llamada si hay parámetros no válidos.
`WebServiceConnector.WSDLURL`	Especifica la URL del archivo WSDL que define la operación de servicio Web.

Resumen de eventos de la clase WebServiceConnector

En la tabla siguiente se enumeran los eventos de la clase WebServiceConnector.

Evento	Descripción
`WebServiceConnector.result`	Se difunde cuando se realiza de forma correcta una llamada a un servicio Web.

Evento	Descripción
WebServiceConnector.send	Se difunde cuando el método trigger() está en curso, después de que se hayan recogido los datos pero antes de que se validen y se inicie la llamada al servicio Web.
WebServiceConnector.status	Se difunde cuando se inicia una llamada a un servicio Web, para informar al usuario del estado de la operación.

WebServiceConnector.multipleSimultaneousAllowed

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.multipleSimultaneousAllowed

Descripción

Propiedad; indica si se pueden realizar varias llamadas al mismo tiempo (true) o no (false). Si esta propiedad es true, la llamada tiene lugar. Si el valor de esta propiedad es false y hay otra llamada en curso, el método WebServiceConnector.trigger() provoca que se emita un evento status con el código CallAlreadyInProgress.

Cuando se están realizando varias llamadas de forma simultánea, no hay ninguna garantía de que finalicen en el mismo orden en el que se han activado. Además, el navegador y/o el sistema operativo pueden limitar el número de operaciones de red simultáneas. El límite más probable que puede encontrar el usuario es que el navegador imponga un número máximo de URL que puedan descargarse de forma simultánea. Esto suele ser configurable en un navegador. Sin embargo, aunque se diera el caso, el navegador pondría en cola los flujos y esto no interferiría con el comportamiento esperado de la aplicación de Flash.

Ejemplo

En el ejemplo siguiente se permite realizar varias llamadas simultáneas a myXmlUrl:

```
myXmlUrl.multipleSimultaneousAllowed = true;
```

WebServiceConnector.operation

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.operation;

Descripción

Propiedad; nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.

Ejemplo

En este ejemplo se devuelven los datos de un servicio Web remoto y se rastrea una sugerencia y el tiempo que tarda el servicio en devolver los datos al archivo SWF. Arrastre un componente WebServiceConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;

var startTime:Number;
var wscListener:Object = new Object();
wscListener.result = function(evt:Object) {
var resultTimeMS:Number = getTimer()-startTime;
trace("result loaded in "+resultTimeMS+" ms.");
trace(evt.target.results);
};
wscListener.send = function(evt:Object) {
startTime = getTimer();
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", wscListener);
wsConn.addEventListener("send", wscListener);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.multipleSimultaneousAllowed = false;
wsConn.trigger();
```

WebServiceConnector.params

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.params

Descripción

Propiedad; especifica datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`. La descripción WSDL del servicio Web determina el tipo de datos.

Cuando se realiza una llamada a métodos de servicio Web, el tipo de datos de la propiedad `params` debe ser un objeto o matriz de ActionScript, como se indica a continuación:

- Si el servicio Web tiene formato de documento, el tipo de datos de `params` es un documento XML.
- Si utiliza el inspector de propiedades o el inspector de componentes para definir la URL y el funcionamiento de WSDL durante la edición, puede proporcionar `params` como matriz de parámetros en el mismo orden que exige el método de servicio Web, por ejemplo `[1, "hello", 2432]`.

Ejemplo

En el ejemplo siguiente se define la propiedad `params` de un componente de servicio Web denominado `wsc`:

```
wsc.params = [param_txt.text];
```

WebServiceConnector.result

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.addEventListener("result", *myListenerObject*)

Descripción

Evento; se difunde cuando se realiza de forma correcta una llamada a un servicio Web.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"result"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)

Puede recuperar el valor de resultado real mediante la propiedad `results`.

Ejemplo

En el ejemplo siguiente se define una función `res` para el evento `result` y se asigna la función al controlador de eventos `addEventListener`:

```
var res = function (ev) {
trace(ev.target.results);
};
wsc.addEventListener("result", res);
```

En este ejemplo se devuelven datos de un servicio Web remoto y se rastrea una sugerencia. Arrastre un componente WebServiceConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;
var wscListener:Object = new Object();
wscListener.result = function(evt:Object) {
  trace(evt.target.results);
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", wscListener);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.trigger();
```

WebServiceConnector.results

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`componentInstance``.results`

Descripción

Propiedad; identifica los datos que se han recibido del servidor como resultado de una operación `trigger()`. Cada componente WebServiceConnector define cómo se recogen estos datos y qué tipos son válidos. Estos datos aparecen cuando la operación RPC ha finalizado correctamente, como señala el evento `result`. Está disponible hasta que el componente se descarga o hasta la siguiente operación RPC.

Los datos devueltos pueden ser voluminosos. Puede gestionar este tamaño de dos maneras:

- Seleccione un clip de película, una línea de tiempo o una pantalla adecuada como elemento principal para el componente WebServiceConnector. La memoria de almacenamiento del componente estará disponible para la eliminación de datos innecesarios cuando se elimine el elemento principal.
- En ActionScript, puede asignar `null` a esta propiedad en cualquier momento.

WebServiceConnector.send

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
componentInstance.addEventListener("send", myListenerObject)
```

Descripción

Evento; se difunde durante el proceso de una operación `trigger()`, después de que se hayan recogido los datos de parámetros pero antes de que se validen y se inicie la llamada al servicio Web. Es una buena ubicación para incluir código que modifique los datos de parámetros antes de la llamada.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"send"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)

Puede recuperar o modificar los valores de parámetro actuales mediante la propiedad `params`.

Ejemplo

En el ejemplo siguiente se define una función `sendFunction` para el evento `send` y se asigna la función al controlador de eventos `addEventListener`:

```
var sendFunction = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
wsc.addEventListener("send", sendFunction);
```

WebServiceConnector.status

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.addEventListener("status", *myListenerObject*)

Descripción

Evento; se difunde cuando se inicia una llamada a un servicio Web, para informar al usuario del estado de la operación.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"status"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)
- `code`: cadena que da el nombre de la condición que se ha producido
- `data`: objeto cuyo contenido depende del código

A continuación se detallan los códigos y los datos asociados disponibles para el evento `status`:

Código	Datos	Descripción
StatusChange	{callsInProgress:nnn}	Este evento se emite siempre que se inicia o finaliza una llamada de servicio Web. El elemento *nnn* es el número de llamadas que se están realizando en este momento.

Código	Datos	Descripción
CallAlreadyInProgress	Sin datos	Este evento se emite si se llama a `trigger()`, `multipleSimultaneousAllowed` es `false` y ya se está realizando una llamada. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento `result` ni `send`.
InvalidParams	Sin datos	Este evento se emite si el método `trigger()` ha detectado que la propiedad `params` no contenía datos válidos. Si el valor de la propiedad `suppressInvalidCalls` es `true`, el intento de llamada se considera realizado y no habrá ningún evento `result` ni `send`.
WebServiceFault	{faultcode: code, faultstring: string, detail: detail}	Este evento se emite si se producen otros problemas durante el proceso de la llamada. Los datos son un objeto SOAPFault. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento `"result"` ni `"send"`. En la siguiente tabla se enumeran los posibles errores.

Estos son los posibles fallos de los servicios Web:

faultcode	faultstring	detail
Timeout	Se ha agotado el tiempo de espera al llamar al método xxx.	
MustUnderstand	No hay ninguna función callback para el encabezado xxx.	
Server.Connection	No se puede conectar con el punto final: xxx	
VersionMismatch	La solicitud implementa la versión xxx y la respuesta implementa la versión yyy.	

faultcode	faultstring	detail
Client.Disconnected	No se ha podido cargar el archivo WSDL.	No se puede cargar el archivo WSDL; si está conectado en este momento, verifique la URI y/o el formato del WSDL xxx.
Server	Formato de WSDL erróneo.	Las definiciones deben ser el primer elemento de un documento WSDL.
Server.NoServicesInWSDL	No se ha podido cargar el archivo WSDL.	No se han encontrado elementos en el archivo WSDL en xxx.
WSDL.UnrecognizedNamespace	El analizador WSDL no tenía ningún documento registrado para el espacio de nombres xxxx.	
WSDL.UnrecognizedBindingName	El analizador WSDL no ha podido encontrar una vinculación denominada xxx en el espacio de nombres yyy.	
WSDL.UnrecognizedPortTypeName	El analizador WSDL no ha podido encontrar un portType denominado xxx en el espacio de nombres yyy.	
WSDL.UnrecognizedMessageName	El analizador WSDL no ha podido encontrar un mensaje denominado xxx en el espacio de nombres yyy.	
WSDL.BadElement	El elemento xxx no se puede resolver.	
WSDL.BadType	El tipo xxx no se puede resolver.	
Client.NoSuchMethod	No se ha podido encontrar el método 'xxx' en el servicio.	

faultcode	faultstring	detail
yyy	yyy - se han notificado errores del servidor; esto depende del servidor con el que se esté comunicando.	
No.WSDLURL.Defined	El componente WebServiceConnector no tenía ninguna URL de WSDL definida.	
Unknown.Call.Failure	La invocación a WebService no se ha podido realizar por motivos desconocidos.	
Client.Disconnected	No se ha podido cargar el esquema importado.	No se puede cargar el esquema; si está conectado en este momento, verifique la URI y/o el formato del esquema en (XXXX).

Ejemplo

En el ejemplo siguiente se define una función `fault` para el evento `status` y se asigna la función al controlador de eventos `addEventListener`. En el ejemplo se escribe intencionadamente de forma incorrecta el URI del servicio para que se devuelva un error de servicio Web (la URL debe ser `"http://www.flash-mx.com/mm/tips/tips.cfc?wsdl"`) y un mensaje donde se pida al usuario que compruebe el URI. Con un componente WebServiceConnector en la biblioteca, añada lo siguiente al primer fotograma de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;

var fault = function (stat) {
  if (stat.code == "WebServiceFault"){
    trace(stat.data.faultcode);
    trace(stat.data.faultstring);
    trace(stat.data.detail);
  }
};

var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("status", fault);
wsConn.WSDLURL = "http://www.flasht-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.trigger();
```

WebServiceConnector.suppressInvalidCalls

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.suppressInvalidCalls

Descripción

Propiedad; indica si debe suprimirse una llamada en caso de que los parámetros no sean válidos. Si su valor es `true`, el método `trigger()` no realizará una llamada si los parámetros vinculados no superan la validación Se emitirá un evento `status` con el código `InvalidParams`. Si esta propiedad es `false`, la llamada tiene lugar con los datos no válidos, del modo requerido.

Ejemplo

En este ejemplo se muestra un error porque los parámetros necesarios no se pasan. Arrastre un componente WebServiceConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
};
var stat:Function = function (error:Object) {
  switch (error.code) {
  case 'InvalidParams' :
    trace("Unable to connect to remote Web Service: "+error.code);
    break;
  case 'StatusChange' :
    break;
  default :
    trace("Error: "+error.code);
    break;
  }
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.addEventListener("status", stat);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
// wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

Para mostrar una sugerencia en lugar de un error, quite la marca de comentario de la línea `wsConn.params = ["Flash"];`.

WebServiceConnector.trigger()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.`trigger();`

Descripción

Método; inicia una llamada a un servicio Web. Cada servicio Web define exactamente lo que esto implica. Si la operación es correcta, el resultado de la operación aparecerá en la propiedad `results` del servicio Web.

El método `trigger()` sigue estos pasos:

1. Si hay datos vinculados a la propiedad `params`, el método ejecuta todas las vinculaciones para garantizar que haya datos actualizados disponibles. Esto también provoca una validación de datos.
2. Si los datos no son válidos y el valor de `suppressInvalidCalls` es `true`, la operación se interrumpe.
3. Si la operación continúa, se emite el evento `send`.
4. La llamada remota real se inicia a través del método de conexión indicado (por ejemplo, HTTP).

Ejemplo

En este ejemplo se devuelven datos de un servicio Web remoto y se rastrea una sugerencia. Arrastre un componente WebServiceConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
};
```

```
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

WebServiceConnector.WSDLURL

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.WSDLURL

Descripción

Propiedad; URL del archivo WSDL que define la operación de servicio Web. Cuando se define esta URL durante la edición, el archivo WSDL se recoge y se analiza de forma inmediata. Los parámetros resultantes y los resultados aparecen en la ficha Esquema del inspector de componentes. La descripción del servicio también aparece en el panel Servicio Web.

Ejemplo

En este ejemplo se devuelven datos de un servicio Web remoto y se rastrea una sugerencia. Arrastre un componente WebServiceConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

CAPÍTULO 58

Componente Window

58

El componente Window muestra el contenido de un clip de película en una ventana provista de una barra de título, un borde y un botón de cierre opcional.

El componente Window puede ser modal o amodal. Una ventana modal impide que las entradas realizadas con el ratón o el teclado se apliquen a otros componentes situados fuera de la ventana. También admite la función de arrastre, es decir, el usuario puede hacer clic en la barra de título y arrastrar la ventana y su contenido a otro lugar. El arrastre de los bordes no cambia el tamaño de la ventana.

La previsualización dinámica de cada instancia de Window refleja los cambios realizados en todos los parámetros (excepto en contentPath) en el inspector de propiedades o en el inspector de componentes durante la edición.

Cuando se añade el componente Window a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.WindowAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte Capítulo 19, “Creación de contenido accesible” en *Utilización de Flash*.

Utilización del componente Window

Puede utilizar una ventana en una aplicación siempre que desee ofrecer al usuario información o una opción que tenga prioridad sobre cualquier otro elemento de la aplicación. Por ejemplo, puede necesitar que un usuario rellene los datos de una ventana de conexión o de una ventana en la que se cambia una contraseña y se confirma una nueva.

Hay varias formas de añadir una ventana a una aplicación. Puede arrastrar una ventana desde el panel Componentes al escenario. También puede llamar a `createClassObject()` (véase `UIObject.createClassObject()`) para añadir una ventana a una aplicación. La tercera forma de añadir ventanas a una aplicación consiste en utilizar la Clase PopUpManager. El uso de Popup Manager permite crear ventanas modales que se superponen a otros objetos del escenario. Para más información, consulte "Clase Window" en la página 1514.

Si se utiliza Popup Manager para añadir un componente Window a un documento, la instancia de Window tendrá su propio Focus Manager, distinto al resto del documento. Si no utiliza Popup Manager, el contenido de la ventana comparte el orden de selección con el resto de los componentes del documento. Para más información sobre el control de la selección, consulte "Clase FocusManager" en la página 745 o "Creación de un desplazamiento personalizado de la selección" en *Utilización de componentes*.

Los componentes como Loader, ScrollPane y Window tienen eventos para determinar que ha finalizado la carga del contenido. Si desea establecer propiedades en el contenido de un componente Loader, ScrollPane o Window, añada la sentencia de propiedad en un controlador de eventos "complete", como se muestra en el siguiente ejemplo:

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._width= 100;
}
my_window.addEventListener("complete", loadtest)
```

Para más información, consulte "Window.complete" en la página 1522.

Parámetros de Window

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Window en el inspector de propiedades o el inspector de componentes (opción de menú Ventana > Inspector de componentes):

closeButton indica si la ventana va a mostrar un botón de cierre (`true`) o no (`false`). Al hacer clic en el botón de cierre se difunde un evento `click`, pero no se cierra la ventana. Debe escribir un controlador que invoque `Window.deletePopUp()` para cerrar la ventana explícitamente. Para más información sobre el evento `click`, consulte `Window.click`.

NOTA	Si una ventana no se creó con Popup Manager, no se podrá cerrar.

contentPath especifica el contenido de la ventana. Puede ser el identificador de vinculación del clip de película o el nombre de símbolo de una pantalla, formulario o diapositiva que incluya el contenido de la ventana. También puede ser una URL absoluta o relativa a un archivo SWF o JPEG que se carga en la ventana. El valor predeterminado es `""`. El contenido cargado se recorta para que quepa en la ventana.

title indica el título de la ventana.

NOTA	Las propiedades minHeight y minWidth se utilizan en rutinas internas de cambio de tamaño. Se definen en UIObject y se sustituyen por distintos componentes según convenga. Estas propiedades pueden utilizarse si se crea un administrador de diseño personalizado en la aplicación. De lo contrario, establecer estas propiedades en el inspector de componentes no produce ningún efecto visible.

A continuación se indican los parámetros adicionales que se pueden definir para cada instancia del componente Window en el inspector de componentes (Ventana > Inspector de componentes):

enabled es un valor booleano que indica si el componente acepta selecciones y entradas. El valor predeterminado es `true`.

visible es un valor booleano que indica si el objeto es visible (`true`) o no (`false`). El valor predeterminado es `true`.

NOTA	Para más información sobre los cinco parámetros de aspecto siguientes, consulte "Utilización de aspectos con el componente Window" en la página 1513.

skinCloseDisabled determina que el botón de cierre está en estado desactivado. El valor predeterminado es `CloseButtonDisabled`.

skinCloseDown determina que el botón de cierre está en estado presionado. El valor predeterminado es `CloseButtonDown`.

skinCloseOver determina el estado del botón de cierre cuando se desplaza el puntero sobre él. El valor predeterminado es `CloseButtonOver`.

skinCloseUp determina que el botón de cierre está en estado sin presionar (predeterminado). El valor predeterminado es `CloseButtonUp`.

skinTitleBackground determina el aspecto de la barra de título. El valor predeterminado es `TitleBackground`.

titleStyleDeclaration asigna el nombre de la declaración de estilo del texto del título. El valor predeterminado es undefined, por lo que la barra de título tiene un texto blanco en negrita. Consulte "Definición de estilos personalizados para grupos de componentes" en *Utilización de componentes*.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente Window utilizando sus propiedades, métodos y eventos. Para más información, consulte "Clase Window" en la página 1514.

Creación de aplicaciones con el componente Window

El procedimiento siguiente explica cómo añadir un componente Window a una aplicación. En este ejemplo, cuando el usuario hace clic en un botón, la ventana muestra una imagen.

Para crear una aplicación con el componente Window:

1. Arrastre una copia del componente Window desde el panel Componentes a la biblioteca del documento actual. Se añadirá el componente a la biblioteca, pero no al escenario.
2. Arrastre un componente de botón desde el panel Componentes al escenario y, en el inspector de propiedades, asígnele el nombre de instancia **my_button**.
3. Abra el panel Acciones e introduzca el controlador de acciones del ratón siguiente en el fotograma 1:

```
/**
 Se requiere:
  - componente Button en el escenario (nombre de instancia: my_button)
  - Componente Window en la biblioteca
*/
import mx.containers.Window;

var my_button:mx.controls.Button;

System.security.allowDomain("http://www.helpexamples.com");

// Crear un objeto detector.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // Crear una instancia de Window.
 var my_win:MovieClip =
  mx.managers.PopUpManager.createPopUp(evt_obj.target, Window, true,
  {title:"Sample Image", contentPath:"http://www.helpexamples.com/
  flash/images/image1.jpg"});
 my_win.setSize(320, 240);
};
// Añadir detector.
my_button.addEventListener("click", buttonListener);
```

En este ejemplo se crea una función `click()` que el detector de eventos `buttonListener` llama cuando el usuario hace clic en el botón **my_button**. El controlador de eventos click, `buttonListener.click()`, llama a `PopUpManager.createPopUp()` para crear una instancia de una ventana que muestre una imagen. Para cerrar la ventana cuando se haga clic en el botón Aceptar o Cancelar, tendrá que escribir otro controlador.

Personalización del componente Window

El componente Window puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()`.

Cuando se cambia el tamaño de la ventana, no se modifica el tamaño del botón de cierre ni del título. El título se alinea a la izquierda y el botón de cierre a la derecha.

Utilización de estilos con el componente Window

Un componente Window admite los siguientes estilos:

Estilo	Tema	Descripción
`themeColor`	Halo	Esquema de colores base de un componente. Los valores posibles son `"haloGreen"`, `"haloBlue"` y `"haloOrange"`. El valor predeterminado es `"haloGreen"`.
`backgroundColor`	Ambos	Color del fondo. El valor predeterminado es blanco para el tema Halo y 0xEFEBEF (gris claro) para el tema Sample.
`borderStyle`	Ambos	El componente Window utiliza una instancia de RectBorder como borde y responde a los estilos definidos en dicha clase. Véase "Clase RectBorder" en la página 1103. El componente Window tiene un estilo de borde específico del componente "`default`" con el tema Halo y "`outset`" con el tema Sample.
`color`	Ambos	Color del texto. El valor predeterminado es 0x0B333C para el tema Halo y en blanco para el tema Sample.
`disabledColor`	Ambos	Color del texto cuando el componente está desactivado. El color predeterminado es 0x848384 (gris oscuro).
`embedFonts`	Ambos	Valor booleano que indica si la fuente especificada en `fontFamily` es una fuente incorporada. Este estilo debe definirse como `true` si `fontFamily` hace referencia a una fuente incorporada. De lo contrario, no se utiliza la fuente incorporada. Si el estilo se define como `true` y `fontFamily` no hace referencia a una fuente incorporada, no se muestra ningún texto. El valor predeterminado es `false`.

Estilo	Tema	Descripción
`fontFamily`	Ambos	Nombre de la fuente del texto. El valor predeterminado es `"_sans"`.
`fontSize`	Ambos	Tamaño de la fuente en puntos. El valor predeterminado es 10.
`fontStyle`	Ambos	Estilo de la fuente: puede ser `"normal"` o `"italic"`. El valor predeterminado es `"normal"`.
`fontWeight`	Ambos	Grosor de la fuente: puede ser `"none"` o `"bold"`. El valor predeterminado es `"none"`. Todos los componentes pueden aceptar además el valor `"normal"` en lugar de `"none"` durante una llamada a `setStyle()`, pero las llamadas posteriores a `getStyle()` devolverán `"none"`.
`textAlign`	Ambos	Alineación del texto: puede ser `"left"`, `"right"` o `"center"`. El valor predeterminado es `"left"`.
`textDecoration`	Ambos	Decoración del texto: puede ser `"none"` o `"underline"`. El valor predeterminado es `"none"`.
`textIndent`	Ambos	Número que indica la sangría del texto. El valor predeterminado es 0.

Los estilos de texto pueden definirse en el propio componente Window o en la declaración de estilo de clase `_global.styles.windowStyles` (sólo estilos de texto y no otros estilos como themeColor o backgroundColor, que proceden de la declaración de estilo de clase `_global.styles.Window`). Esto tiene la ventaja de que no crea configuraciones de estilo que se propaguen a los componentes secundarios mediante la herencia de estilo.

En el siguiente ejemplo se muestra cómo poner en cursiva el título de un componente Window sin que esta configuración se propague a los componentes secundarios.

```
import mx.containers.Window;
_global.styles.windowStyles.setStyle("fontStyle", "italic");
createClassObject(Window, "window", 1, {title: "A Window"});
```

Observe que en este ejemplo se define la propiedad antes de crear instancias de ventana mediante `createClassObject()`. Para que los estilos tengan efecto, deben definirse antes de crear la ventana.

Utilización de aspectos con el componente Window

El componente Window utiliza aspectos para el fondo de título y el botón de cierre y una instancia de RectBorder para el borde. Los aspectos de Window se encuentran en la carpeta Flash UI Components 2/Themes/ MMDefault/Window Assets de cada archivo de tema. Para más información sobre la aplicación de aspectos, consulte "Aplicación de aspectos a los componentes" en *Utilización de componentes*. Para más información sobre la clase RectBorder y su utilización para personalizar bordes, consulte "Clase RectBorder" en la página 1103.

El aspecto del fondo de título se muestra siempre. La altura del fondo la determinan los gráficos de aspecto. La anchura del aspecto la define el componente Window de acuerdo con el tamaño de la instancia de Window. Los aspectos de cierre se muestran cuando la propiedad `closeButton` está establecida en `true` y cuando se produce un cambio de estado por la interacción del usuario.

Un componente Window utiliza las propiedades de aspecto siguientes:

Propiedad	Descripción
`skinTitleBackground`	Barra de título. El valor predeterminado es `TitleBackground`.
`skinCloseUp`	Botón de cierre. El valor predeterminado es `CloseButtonUp`.
`skinCloseDown`	Botón de cierre en estado presionado. El valor predeterminado es `CloseButtonDown`.
`skinCloseDisabled`	Botón de cierre en estado desactivado. El valor predeterminado es `CloseButtonDisabled`.
`skinCloseOver`	Botón de cierre cuando el puntero pasa sobre él. El valor predeterminado es `CloseButtonOver`.

En el siguiente ejemplo se muestra cómo crear un símbolo de clip de película nuevo para utilizarlo como fondo del título.

Para definir el título de un componente Window en un símbolo de clip de película personalizado:

1. Cree un nuevo archivo FLA.
2. Cree un nuevo símbolo seleccionando Insertar > Nuevo símbolo.
3. Asígnele el nombre `TitleBackground`.
4. Si no se muestra la vista avanzada, haga clic en el botón Avanzado.
5. Seleccione Exportar para ActionScript.
6. El identificador se rellenará automáticamente con `TitleBackground`.

7. Establezca `mx.skins.SkinElement` como clase de AS 2.0.

   SkinElement es una clase simple que puede utilizarse para todos los elementos de aspecto que no proporcionan su propia implementación de ActionScript. Proporciona el movimiento y las funciones de cambio de tamaño que requiere el marco de componentes de la versión 2 de la arquitectura de componentes de Macromedia.
8. Verifique que Exportar en primer fotograma esté seleccionado y haga clic en Aceptar.
9. Abra el símbolo nuevo para editarlo.
10. Utilice las herramientas de dibujo para crear un cuadro con relleno en rojo y línea negra.
11. Establezca un estilo de borde muy fino.
12. Defina el cuadro, incluido el borde, para que se posicione en (0,0) y tenga una anchura de 100 y una altura de 22.

    El componente Window definirá la anchura adecuada del aspecto, pero utilizará la altura existente como altura del título.
13. Haga clic en el botón Atrás para volver a la línea de tiempo principal.
14. Arrastre un componente Window al escenario.
15. Seleccione Control > Probar película.

Clase Window

Herencia MovieClip > Clase UIObject > Clase UIComponent > View > ScrollView > Window

Nombre de clase de ActionScript mx.containers.Window

Las propiedades de la clase Window permiten realizar lo siguiente en tiempo de ejecución: definir el título, añadir un botón de cierre y definir el contenido que se muestra. Si una propiedad de la clase Window se define con ActionScript, sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

La mejor manera de crear una instancia de Window es llamando a `PopUpManager.createPopUp()`. Este método crea una ventana que puede ser modal (que solapa y desactiva los objetos existentes de la aplicación) o amodal. Por ejemplo, el código siguiente crea una instancia de Window modal (el último parámetro indica modalidad):

```
var newWindow = PopUpManager.createPopUp(this, Window, true);
```

Flash simula la modalidad creando una ventana transparente grande por debajo del componente Window. Debido a la forma en la que se representan las ventanas transparentes, es posible que advierta una ligera atenuación de los objetos situados debajo. Para definir una transparencia efectiva, hay que cambiar el valor de `_global.style.modalTransparency` de 0 (completamente transparente) a 100 (opaco). Si hace que la ventana sea parcialmente transparente, también podrá definir el color de la ventana cambiando el aspecto Modal del tema predeterminado.

Si utiliza `PopUpManager.createPopUp()` para crear una ventana modal, deberá llamar a `Window.deletePopUp()` para eliminarlo y eliminar también la ventana transparente. Por ejemplo, si utiliza el botón de cierre de la ventana, deberá escribir el código siguiente:

```
var win = PopUpManager.createPopUp(_root, Window, true,
  {closeButton:true});
function click(evt){
  evt.target.deletePopUp();
}
win.addEventListener("click", this);
```

La ejecución del código no se interrumpe cuando se crea una ventana modal. En otros entornos, como Microsoft Windows, si crea una ventana modal, las líneas de código siguientes a la creación de la ventana no se ejecutan hasta que ésta se cierra. En Flash, las líneas de código siguen ejecutándose después de crear la ventana y antes de que se cierre.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a esta propiedad, utilice el código siguiente:

```
trace(mx.containers.Window.version);
```

NOTA

El código `trace(myWindowInstance.version);` devuelve `undefined`.

Resumen de métodos de la clase Window

En la tabla siguiente se muestra el método de la clase Window.

Método	Descripción
`Window.deletePopUp()`	Elimina una instancia de Window creada por `PopUpManager.createPopUp()`.

Métodos heredados de la clase UIObject

En la tabla siguiente se enumeran los métodos que hereda la clase Window de la clase UIObject. Al llamar a estos métodos desde el objeto Window, debe utilizarse la forma *WindowInstance.methodName*.

Método	Descripción
UIObject.createClassObject()	Crea un objeto en la clase especificada.
UIObject.createObject()	Crea un subobjeto en un objeto.
UIObject.destroyObject()	Elimina una instancia de componente.
UIObject.doLater()	Llama a una función cuando se han establecido parámetros en el inspector de propiedades y el inspector de componentes.
UIObject.getStyle()	Obtiene la propiedad de estilo de la declaración de estilo o del objeto.
UIObject.invalidate()	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
UIObject.move()	Mueve el objeto a la posición indicada.
UIObject.redraw()	Fuerza la validación del objeto, de forma que se pueda dibujar sobre el fotograma actual.
UIObject.setSize()	Cambia el tamaño del objeto al indicado.
UIObject.setSkin()	Define un aspecto en el objeto.
UIObject.setStyle()	Define la propiedad de estilo en la declaración de estilo o en el objeto.

Métodos heredados de la clase UIComponent

En la tabla siguiente se enumeran los métodos que hereda la clase Window de la clase UIComponent. Al llamar a estos métodos desde el objeto Window, debe utilizarse la forma *WindowInstance.methodName*.

Método	Descripción
UIComponent.getFocus()	Devuelve una referencia al objeto seleccionado.
UIComponent.setFocus()	Define la selección en la instancia de componente.

Resumen de propiedades de la clase Window

En la tabla siguiente se enumeran las propiedades de la clase Window.

Propiedad	Descripción
`Window.closeButton`	Indica si la barra de título incluye un botón de cierre (`true`) o no (`false`).
`Window.content`	Referencia al contenido (clip de película raíz) de la ventana (sólo lectura).
`Window.contentPath`	Define el nombre del contenido que ha de aparecer en la ventana.
`Window.title`	Texto que aparece en la barra de título.
`Window.titleStyleDeclaration`	Declaración de estilos que asigna formato al texto de la barra de título.

Propiedades heredadas de la clase UIObject

En la tabla siguiente se enumeran las propiedades que hereda la clase Window de la clase UIObject. Al acceder a estas propiedades desde el objeto Window, debe utilizarse la forma *`WindowInstance.propertyName`*.

Propiedad	Descripción
`UIObject.bottom`	Sólo lectura; posición del borde inferior del objeto con respecto al borde inferior de su elemento principal correspondiente.
`UIObject.height`	Sólo lectura; altura del objeto, expresada en píxeles.
`UIObject.left`	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
`UIObject.right`	Sólo lectura; posición del borde derecho del objeto con respecto al borde derecho de su elemento principal correspondiente.
`UIObject.scaleX`	Número que indica el factor de escala en la dirección *x* del objeto con respecto a su elemento principal correspondiente.
`UIObject.scaleY`	Número que indica el factor de escala en la dirección *y* del objeto con respecto a su elemento principal correspondiente.
`UIObject.top`	Sólo lectura; posición del borde superior del objeto con respecto a su elemento principal correspondiente.
`UIObject.visible`	Valor booleano que indica si el objeto es visible (`true`) o no (`false`).

Propiedad	Descripción
UIObject.width	Sólo lectura; anchura del objeto, expresada en píxeles.
UIObject.x	Sólo lectura; borde izquierdo del objeto, expresado en píxeles.
UIObject.y	Sólo lectura; borde superior del objeto, expresado en píxeles.

Propiedades heredadas de la clase UIComponent

En la tabla siguiente se enumeran las propiedades que hereda la clase Window de la clase UIComponent. Al acceder a estas propiedades desde el objeto Window, debe utilizarse la forma *WindowInstance.propertyName*.

Propiedad	Descripción
UIComponent.enabled	Indica si el componente puede recibir selecciones y entradas.
UIComponent.tabIndex	Número que indica el orden de tabulación para un componente de un documento.

Resumen de eventos de la clase Window

En la tabla siguiente se enumeran los eventos de la clase Window.

Evento	Descripción
Window.click	Se difunde cuando se hace clic en (se suelta) el botón de cierre.
Window.complete	Se difunde cuando se crea una ventana.
Window.mouseDownOutside	Se difunde cuando se hace clic con el botón del ratón (se suelta) fuera de la ventana modal.

Eventos heredados de la clase UIObject

En la tabla siguiente se enumeran los eventos que hereda la clase Window de la clase UIObject.

Evento	Descripción
UIObject.draw	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
UIObject.hide	Se difunde cuando el estado de un objeto pasa de ser visible a invisible.

Evento	Descripción
UIObject.load	Se difunde cuando se crean subobjetos.
UIObject.move	Se difunde cuando se mueve el objeto.
UIObject.resize	Se difunde cuando cambia el tamaño de un objeto.
UIObject.reveal	Se difunde cuando el estado de un objeto pasa de ser invisible a visible.
UIObject.unload	Se difunde durante la descarga de los subobjetos.

Eventos heredados de la clase UIComponent

En la tabla siguiente se enumeran los eventos que hereda la clase Window de la clase UIComponent.

Evento	Descripción
UIComponent.focusIn	Se difunde cuando se selecciona un objeto.
UIComponent.focusOut	Se difunde cuando un objeto deja de seleccionarse.
UIComponent.keyDown	Se difunde cuando se presiona una tecla.
UIComponent.keyUp	Se difunde cuando se suelta una tecla.

Window.click

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObject:Object) {
  // ...
};
windowInstance.addEventListener("click", listenerObject);
```

Sintaxis 2:

```
on (click) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) sobre el botón de cierre.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*windowInstance*) distribuye un evento (en este caso, `click`) y éste se controla mediante una función, denominada también *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de Window. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia de Window `myWindow`, envía "_level0.myWindow" al panel Salida:

```
on(click){
  trace(this);
}
```

Ejemplo

En el siguiente ejemplo se crea una ventana modal con un botón de cierre. Se define un controlador de acciones del ratón que llama al método `click()` para eliminar la ventana cuando el usuario haga clic en el botón. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");
```

```
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function() {
 my_win.deletePopUp();
};
my_win.addEventListener("click", winListener);
```

Véase también

EventDispatcher.addEventListener(), Window.closeButton

Window.closeButton

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.closeButton

Descripción

Propiedad; valor booleano que indica si la barra de título debe llevar un botón de cierre (`true`) o no (`false`). La propiedad debe definirse en el parámetro *initObject* del método PopUpManager.createPopUp(). El valor predeterminado es `false`.

Al hacer clic en el botón de cierre se difunde un evento `click`, pero no se cierra la ventana. Debe escribir un controlador que invoque a Window.deletePopUp() para cerrar la ventana explícitamente. Para más información sobre el evento `click`, consulte Window.click.

Ejemplo

En el siguiente ejemplo se crea una ventana emergente y se define la propiedad `closeButton` para añadirle un botón de cierre. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
```

```
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
```

Véase también

`PopUpManager.createPopUp()`, `Window.click`

Window.complete

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

```
listenerObject = new Object();
listenerObject.complete = function(eventObject){
  ...
}
windowInstance.addEventListener("complete", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se crea una ventana. Utilice este evento para adaptar el tamaño de una ventana a su contenido.

Una instancia de componente (*`windowInstance`*) distribuye un evento (en este caso, `complete`) y éste se controla mediante una función, denominada también *controlador*, asociada con el objeto detector (*`listenerObject`*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*`eventObject`*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte "Clase EventDispatcher" en la página 515.

Ejemplo

En el siguiente ejemplo, se crea una ventana y, a continuación, se define un controlador `complete` que adapta el tamaño de la ventana a su contenido. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 my_win.deletePopUp();
};
winListener.complete = function(evt_obj:Object) {
 my_win.setSize(my_win.content._width, my_win.content._height + 25);
}
my_win.addEventListener("click", winListener);
my_win.addEventListener("complete", winListener);
```

Véase también

`EventDispatcher.addEventListener()`

Window.content

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.content

Descripción

Propiedad de sólo lectura; referencia al contenido (clip de película raíz) de la ventana. Esta propiedad devuelve un objeto MovieClip. Cuando se asocia un símbolo de la biblioteca, el valor predeterminado es una instancia del símbolo asociado. Si se carga el contenido de una URL, el valor predeterminado es `undefined` hasta que comienza la operación de carga.

Ejemplo

En el siguiente ejemplo, se crea una ventana y, a continuación, se define un controlador `complete` que adapta el tamaño de la ventana a su contenido. Se utiliza la propiedad `content` para hacer referencia a la anchura del contenido del clip de película de la ventana. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 my_win.deletePopUp();
};
winListener.complete = function(evt_obj:Object) {
 my_win.setSize(my_win.content._width, my_win.content._height + 25);
}
my_win.addEventListener("click", winListener);
my_win.addEventListener("complete", winListener);
```

Window.contentPath

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.contentPath

Descripción

Propiedad; define el nombre del contenido que ha de aparecer en la ventana. Este valor puede ser el identificador de vinculación de un clip de película de la biblioteca o la URL absoluta o relativa de un archivo SWF o JPEG que se carga. El valor predeterminado es "" (una cadena vacía).

Ejemplo

En el siguiente ejemplo se crea una ventana y se utiliza la propiedad `contentPath` para especificar la ubicación de la imagen que se mostrará en la ventana. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

// Crear ventana.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true, {
  contentPath:"http://www.flash-mx.com/images/image2.jpg"});
```

Window.deletePopUp()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.deletePopUp()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina la instancia de Window y quita el estado modal. Este método sólo se puede llamar para instancias de Window creadas por medio de `PopUpManager.createPopUp()`.

Ejemplo

En el ejemplo siguiente se crea una ventana modal y se define un controlador de acciones del ratón que llama a la función `deletePopUp()` para eliminar la ventana. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function() {
 my_win.deletePopUp();
};
my_win.addEventListener("click", winListener);
```

Window.mouseDownOutside

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
var listenerObject:Object = new Object();
listenerObject.mouseDownOutside = function(eventObject:Object) {
  // ...
};
windowInstance.addEventListener("mouseDownOutside", listenerObject);
```

Sintaxis 2:

```
on (mouseDownOutside) {
  // ...
}
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) fuera de la ventana modal. Este evento se utiliza muy raramente, pero puede servir para descartar una ventana si el usuario trata de interactuar con algún elemento situado fuera de la misma.

El primer ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*windowInstance*) distribuye un evento (en este caso, `mouseDownOutside`) y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Se define un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al método del objeto detector. El objeto de evento tiene propiedades que contienen información sobre el evento. Estas propiedades sirven para escribir el código que controla el evento. Finalmente, se llama al método `EventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información, consulte “Clase EventDispatcher” en la página 515.

El segundo ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia de Window. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia de Window `myWindowComponent`, envía "_level0.myWindowComponent" al panel Salida:

```
on (mouseDownOutside) {
  trace(this);
}
```

Ejemplo

En el ejemplo siguiente se crea una instancia de Window y se define un controlador `mouseDownOutside` que muestra un mensaje si el usuario hace clic fuera de la ventana. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

// Crear ventana.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  undefined, true);

// Crear un objeto detector.
var winListener:Object = new Object();
winListener.mouseDownOutside = function(evt_obj:Object)
{
    trace("mouseDownOutside event triggered.");
}
// Añadir detector.
my_win.addEventListener("mouseDownOutside", winListener);
```

Véase también

`EventDispatcher.addEventListener()`

Window.title

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.title

Descripción

Propiedad; cadena que indica el texto de la barra de título. El valor predeterminado es "" (una cadena vacía).

Ejemplo

En el siguiente ejemplo se crea una ventana emergente y se utiliza la propiedad `title` para establecer como título "Hello World". Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1:

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.managers.PopUpManager
import mx.containers.Window

// Crear ventana.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true);

// Definir atributos de la ventana.
my_win.title = "Hello World!";
my_win.setSize(200, 100);
my_win.move(20, 20);
```

Window.titleStyleDeclaration

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX 2004.

Sintaxis

windowInstance.titleStyleDeclaration

Descripción

Propiedad; cadena que indica la declaración de estilos que asigna formato a la barra de título de una ventana. El valor predeterminado es `undefined`, que significa texto blanco en negrita.

Ejemplo

En el siguiente ejemplo se crea una declaración de estilo CSS para que el texto tenga un tamaño de 14 puntos y aparezca en cursiva y subrayado. Se utiliza la propiedad `titleStyleDeclaration` para aplicar dicho estilo al título de la ventana emergente que crea. Debe arrastrar un componente Window desde el panel Componentes hasta la biblioteca del documento actual y, a continuación, añadir el siguiente código al fotograma 1.

```
/**
 Se requiere:
  - Componente Window en la biblioteca
*/

import mx.styles.CSSStyleDeclaration
import mx.managers.PopUpManager
import mx.containers.Window

// Crear una nueva CSSStyleDeclaration denominada TitleStyles
// e incluirla en la lista de estilos globales.
_global.styles.TitleStyles = new CSSStyleDeclaration();

// Personalizar estilos.
_global.styles.TitleStyles.fontStyle = "italic";
_global.styles.TitleStyles.textDecoration = "underline";
_global.styles.TitleStyles.color = 0xff0000;
_global.styles.TitleStyles.fontSize = 14;

// Crear ventana.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, titleStyleDeclaration:"TitleStyles"});

// Definir atributos de la ventana.
my_win.title = "Testing Styles";
my_win.setSize(200, 100);
my_win.move(20, 20);

// Crear un objeto detector.
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 trace("closing window");
 evt_obj.target.deletePopUp();
};
// Añadir detector.
my_win.addEventListener("click", winListener);
```

Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en *Utilización de componentes*.

CAPÍTULO 59

Componente XMLConnector (sólo en Flash Professional)

59

El componente XMLConnector permite leer o escribir documentos XML mediante operaciones HTTP `GET` y `POST`. Actúa como conector entre otros componentes y orígenes de datos XML externos. El componente XMLConnector se comunica con otros componentes de la aplicación mediante código ActionScript o funciones de vinculación de datos del entorno de edición de Flash. El componente XMLConnector tiene propiedades, métodos y eventos, pero no aparece en pantalla en tiempo de ejecución.

Utilización del componente XMLConnector (sólo en Flash Professional)

El componente XMLConnector proporciona a su aplicación acceso a cualquier origen de datos externo que devuelve o recibe XML a través de HTTP. La forma más fácil de conectarse con un origen de datos XML externo y utilizar los parámetros y resultados de ese origen de datos para su aplicación es especificar un *esquema*, la estructura del documento XML que identifica los elementos de datos del documento al que quiere vincularse.

Para más información sobre el trabajo con el componente XMLConnector, consulte "Conexión a datos XML con el componente XMLConnector (sólo para Flash Professional)" en *Utilización de Flash*.

Parámetros de XMLConnector

A continuación se indican los parámetros de edición que se pueden definir para cada instancia de componente XMLConnector en la ficha Parámetros del inspector de componentes:

URL es una cadena que apunta a un origen de datos XML externo.

direction es una cadena que define qué operación HTTP hay que realizar cuando se llama al método `XMLConnector.trigger()`. Este parámetro puede tener el valor `"send"`, `"receive"` o `"send/receive"`.

Un valor `"send"` significa que los datos XML se envían (vía HTTP POST) a la URL, pero Flash omite cualquier dato que se devuelva. No se establece ningún valor para la propiedad `XMLConnector.results` y no hay ningún evento `result`.

Un valor `"receive"` significa que no se envían datos XML a la URL. Flash accede a la URL vía HTTP GET y espera que se devuelvan datos XML válidos.

Un valor `"send/receive"` significa que Flash envía los datos XML vía HTTP POST y espera que se devuelvan datos XML válidos.

Si el valor del parámetro `direction` es `null` o unrecognized, el valor predeterminado es `"send/receive"`.

ignoreWhite es un valor booleano; el valor predeterminado es `false`. Cuando se establece este parámetro en `true`, los nodos de texto que contienen sólo espacio en blanco se descartan durante el proceso de análisis. Los nodos de texto con espacio en blanco al principio o al final no se ven afectados.

multipleSimultaneousAllowed es un valor booleano; cuando tiene el valor de `true`, permite que se inicie una operación `trigger()` cuando ya hay una operación `trigger()` en curso. Si hay varias operaciones `trigger()` simultáneas es posible que no se completen en el mismo orden en que se llamaron. Además, Flash Player puede limitar el número de operaciones de red simultáneas. Este límite varía para cada versión y plataforma. Cuando el parámetro se establece en `false`, no puede iniciarse una operación `trigger()` si hay otra en curso.

suppressInvalidCall es un valor booleano; cuando se establece en `true`, suprime la operación `trigger()` si los parámetros de datos no son válidos. Cuando suppressInvalidCall se establece en `false`, se ejecuta la operación `trigger()` y, si es necesario, utiliza datos no válidos.

Flujo de trabajo normal del componente XMLConnector

En el siguiente procedimiento se muestra el flujo de trabajo típico del componente XMLConnector.

Para utilizar un componente XMLConnector:

1. Añada una instancia del componente XMLConnector a su aplicación y asígnele un nombre de instancia.
2. Utilice la ficha Parámetros del inspector de componentes para introducir la URL del origen de datos XML externo al que desee acceder.
3. Utilice la ficha Esquema del inspector de componentes para especificar un esquema para el documento XML.

NOTA	Puede utilizar el botón de importación de esquema de muestra para automatizar este proceso.

4. Utilice la ficha Vinculaciones del inspector de componentes para vincular elementos de datos (`params` y `results`) del documento XML a las propiedades de los componentes visuales de su aplicación.

   Por ejemplo, puede conectarse a un documento XML que proporcione información meteorológica y vincular los elementos de datos Ubicación y Temperatura a los componentes Label de su aplicación. El nombre y la temperatura de una ciudad concreta aparecen en la aplicación en tiempo de ejecución.
5. Utilice un desencadenante para iniciar la operación de vinculación de datos con uno de los siguientes métodos:
   - Asocie el comportamiento Trigger Data Source a un botón.
   - Utilice su propio código ActionScript para llamar al método `trigger()` en el componente XMLConnector.
   - Cree una vinculación entre el parámetro XML y un control de interfaz de usuario y establezca su propiedad Kind en AutoTrigger. Para más información, consulte “Tipos de esquema” en *Utilización de Flash*.

Para ver un ejemplo paso a paso que conecte y muestre XML mediante el componente XMLConnector, consulte el “XML Tutorial: Timesheet” de los tutoriales de integración de datos en www.macromedia.com/go/data_integration.

Clase XMLConnector (sólo en Flash Professional)

Herencia RPCCall > XMLConnector

Nombre de clase de ActionScript mx.data.components.XMLConnector

La clase XMLConnector le permite enviar o recibir archivos XML mediante HTTP. Puede utilizar código ActionScript para vincular otros componentes a un origen de datos que devuelva datos XML, permitiendo así la comunicación entre los componentes.

Resumen de métodos de la clase XMLConnector

En la tabla siguiente se muestra el método de la clase XMLConnector.

Método	Descripción
`XMLConnector.trigger()`	Inicia una llamada a procedimiento remoto.

Resumen de propiedades de la clase XMLConnector

En la tabla siguiente se enumeran las propiedades de la clase XMLConnector.

Propiedad	Descripción
`XMLConnector.direction`	Indica si los datos se están enviando, recibiendo o ambas cosas.
`XMLConnector.ignoreWhite`	Indica si los nodos de texto que contienen sólo espacio en blanco se descartan durante el proceso de análisis.
`XMLConnector.multipleSimultaneousAllowed`	Indica si se pueden realizar varias llamadas a la vez.
`XMLConnector.params`	Especifica los datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`.
`XMLConnector.results`	Identifica los datos que se han recibido del servidor como resultado de la operación `trigger()`.
`XMLConnector.suppressInvalidCalls`	Indica si debe suprimirse una llamada si hay parámetros no válidos.
`XMLConnector.URL`	URL que utiliza el componente en operaciones HTTP.

Resumen de eventos de la clase XMLConnector

En la tabla siguiente se enumeran los eventos de la clase XMLConnector.

Evento	Descripción
`XMLConnector.result`	Se difunde cuando una llamada a procedimiento remoto se realiza correctamente.
`XMLConnector.send`	Se difunde cuando hay un método `trigger()` en marcha, después de que los datos de parámetros se hayan recopilado pero antes de que se validen y se inicie la llamada a procedimiento remoto.
`XMLConnector.status`	Se difunde cuando se inicia una llamada a procedimiento remoto para informar al usuario del estado de la operación.

XMLConnector.direction

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.direction

Descripción

Propiedad; indica si los datos se están enviando, recibiendo o ambas cosas. Los valores son los siguientes:

- `send` los datos XML de la propiedad `params` se envían a través del método HTTP `POST` a la URL del documento XML. Los datos que se devuelven se omiten. No se establece ningún valor para la propiedad `results` y no hay ningún evento `result`.

  Nota: las propiedades `params` y `results` y el evento `result` se heredan de la interfaz API del componente RPC.

- `receive` No se envían datos de `params` a la URL. Se accede a la URL del documento XML a través de HTTP `GET` y se espera recibir datos XML válidos de la URL.
- `send/receive` Los datos de `params` se envían a la URL y se espera recibir datos XML válidos de la URL.

Ejemplo

En el ejemplo siguiente se establece el valor `receive` para la dirección del documento `mysettings.xml`:

```
myXMLConnector.direction = "receive";
myXMLConnector.URL = "mysettings.xml";
myXMLConnector.trigger();
```

XMLConnector.ignoreWhite

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.ignoreWhite

Descripción

Propiedad; valor booleano. Cuando se establece este parámetro en `true`, los nodos de texto que contienen sólo espacio en blanco se descartan durante el proceso de análisis. Los nodos de texto con espacio en blanco al principio o al final no se ven afectados. El valor predeterminado es `false`.

Ejemplo

El código siguiente establece la propiedad `ignoreWhite` en `true`:

```
myXMLConnector.ignoreWhite = true;
```

XMLConnector.multipleSimultaneousAllowed

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.multipleSimultaneousAllowed

Descripción

Propiedad; indica si se pueden realizar varias llamadas al mismo tiempo. Si esta propiedad es `false`, el método `XMLConnector.trigger()` realiza una llamada si ya se está realizando otra llamada. Se emitirá un evento `status` con el código `CallAlreadyInProgress`. Si esta propiedad es `true`, la llamada tiene lugar.

Cuando se están realizando varias llamadas de forma simultánea, no hay ninguna garantía de que finalicen en el mismo orden en el que se han activado. Además, el navegador y/o el sistema operativo pueden limitar el número de operaciones de red simultáneas. El límite más probable que puede encontrar el usuario es que el navegador imponga un número máximo de URL que puedan descargarse de forma simultánea. Esto suele ser configurable en un navegador. Sin embargo, aunque se diera el caso, el navegador pondría en cola los flujos y esto no interferiría con el comportamiento esperado de la aplicación de Flash.

Ejemplo

En este ejemplo se recupera un archivo XML remoto mediante el componente XMLConnector al establecer la propiedad `direction` en `receive`. Arrastre un componente XMLConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

En este ejemplo se especifica la URL del archivo XML y se establece `multipleSimultaneousAllowed` en `false`. Activa la instancia de XMLConnector tres veces, lo que hace que el método `status` del detector de eventos muestre el código de error `CallAlreadyInProgress` dos veces en el panel Salida. El primer intento se envía correctamente desde Flash al servidor. Cuando la primera activación recibe correctamente un resultado, se difunde un evento `result` y el paquete XML recibido se muestra en el panel Salida.

XMLConnector.params

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.params

Descripción

Propiedad; especifica datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger`(). Cada componente RPC define cómo se utilizan estos datos y qué tipos son válidos.

Ejemplo

En el ejemplo siguiente se definen los parámetros `name` y `city` de `myXMLConnector`:

```
myXMLConnector.params = new XML("<mydoc><name>Bob</name><city>Oakland</
  city></mydoc>");
```

XMLConnector.result

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.addEventListener("result", *myListenerObject*)

Descripción

Evento; se difunde cuando una llamada a procedimiento remoto se realiza correctamente.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"result"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)

Puede recuperar el valor de resultado real mediante la propiedad `results`.

Ejemplo

En el ejemplo siguiente se define una función `res` para el evento `result` y se asigna la función al controlador de eventos `addEventListener`:

```
var res = function (ev) {
trace(ev.target.results);
};
xcon.addEventListener("result", res);
```

XMLConnector.results

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`componentInstance``.results`

Descripción

Propiedad; identifica los datos que se han recibido del servidor como resultado de una operación `trigger()`. Cada componente RPC define cómo se recogen estos datos y qué tipos son válidos. Estos datos aparecen cuando la operación RPC ha finalizado correctamente, como señala el evento `result`. Está disponible hasta que el componente se descarga o hasta la siguiente operación RPC.

Es posible que los datos que se devuelvan sean de gran tamaño. Puede gestionarse de dos maneras:

- Seleccione un clip de película, una línea de tiempo o una pantalla adecuada como elemento principal para el componente RPC. La memoria del componente estará disponible para la eliminación de datos innecesarios cuando se elimine el elemento principal.
- En ActionScript, puede asignar `null` a esta propiedad en cualquier momento.

Ejemplo

En el ejemplo siguiente se rastrea la propiedad `results` de `myXMLConnector`:

```
trace(myXMLConnector.results);
```

XMLConnector.send

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

`componentInstance``.addEventListener("send", `*`myListenerObject`*`)`

Descripción

Evento; se difunde cuando hay una operación `trigger()` en curso, después de que los datos de parámetros se hayan recopilado pero antes de que se validen y se inicie la llamada a procedimiento remoto. Es una buena ubicación para incluir código que modifique los datos de parámetros antes de la llamada.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"send"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)

Puede recuperar o modificar los valores de parámetro actuales mediante la propiedad `params`.

Ejemplo

En el ejemplo siguiente se define una función `sendFunction` para el evento `send` y se asigna la función al controlador de eventos `addEventListener`:

```
var sendFunction = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
xcon.addEventListener("send", sendFunction);
```

XMLConnector.status

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

```
componentInstance.addEventListener("status", myListenerObject)
```

Descripción

Evento; se difunde cuando se inicia una llamada a procedimiento remoto para informar al usuario del estado de la operación.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena `"status"`
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)
- `code`: cadena que da el nombre de la condición específica que se ha producido
- `data`: objeto cuyo contenido depende del código

Se establece el valor Fault en el campo de código del evento Status si se producen problemas con la llamada, de la manera siguiente:

Código	Datos	Descripción
Fault	{faultcode: code, faultstring: string, detail: detail, element: element, faultactor: actor}	Este evento se emite si se producen otros problemas durante el proceso de la llamada. Los datos son un objeto SOAPFault. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento `result` ni `send`.

A continuación se indican los fallos que se pueden producir con el evento `status`:

Código de fallo	Cadena de fallo	Notas
`XMLConnector.Not.XML`	params no es un objeto XML	El valor `params` debe ser un objeto XML de ActionScript.
`XMLConnector.Parse.Error`	params tenía el error NN de análisis XML	La propiedad `status` del objeto XML `params` tenía un valor distinto de cero NN. Para ver los posibles errores NN, consulte `XML.status` en *Referencia del lenguaje ActionScript 2.0*.

Código de fallo	Cadena de fallo	Notas
`XMLConnector.No.Data.Received`	no se han recibido datos del servidor	Debido a diversas limitaciones de navegador, este mensaje puede significar: (a) que la URL del servidor no era válida, no respondía o ha devuelto un código de error HTTP; o (b) que la solicitud del servidor se ha ejecutado correctamente pero la respuesta generada tiene 0 bytes de datos. Para evitar esta restricción, diseñe su aplicación de manera que el servidor nunca devuelva 0 bytes de datos. Si recibe el código de fallo `XMLConnector.No.Data.Received`, sabrá que ha habido un error de servidor y podrá notificárselo al usuario de la forma correspondiente.
`XMLConnector.Results.Parse.Error`	los datos recibidos tenían un error NN de análisis XML	El código XML recibido no era válido, como ha determinado el analizador XML integrado de Flash Player. Para ver los posibles errores NN, consulte `XML.status` en *Referencia del lenguaje ActionScript 2.0*.
`XMLConnector.Params.Missing`	la dirección es 'send' o 'send/receive', pero params es null	

Ejemplo

En el ejemplo siguiente se define una función `statusFunction` para el evento `status` y se asigna la función al controlador de eventos `addEventListener`:

```
var statusFunction = function (stat) {
trace(stat.code);
trace(stat.data.faultcode);
trace(stat.data.faultstring);
};
xcon.addEventListener("status", statusFunction);
```

XMLConnector.suppressInvalidCalls

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.suppressInvalidCalls

Descripción

Propiedad; indica si debe suprimirse una llamada en caso de que los parámetros no sean válidos. Si su valor es `true`, el método `trigger()` no realizará una llamada si los parámetros vinculados no superan la validación Se emitirá un evento `status` con el código `InvalidParams`. Si esta propiedad es `false`, la llamada tiene lugar con los datos no válidos, del modo requerido.

Ejemplo

En este ejemplo se muestra un error porque los parámetros necesarios no se pasan. Arrastre un componente XMLConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  switch (evt.code) {
  case 'Fault' :
    trace("ERROR! ["+evt.data.faultcode+"]");
    trace("\t"+evt.data.faultstring);
    break;
  case 'InvalidParams' :
    trace("ERROR! ["+evt.code+"]");
    break;
  }
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
```

```
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "send/receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/login_xml.cfm";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = false;
// myXMLConnector.params = new XML("<login username='Mort'
  password='Guacamole' />");
myXMLConnector.trigger();
```

Elimine los comentarios de la penúltima línea de código para que el fragmento funcione correctamente.

XMLConnector.trigger()

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.trigger()

Descripción

Método; inicia una llamada a procedimiento remoto mediante el componente XMLConnector. Puede ser tanto obteniendo como enviando el archivo XML especificado. Si la operación es correcta, el resultado de la operación aparecerá en la propiedad `results` del componente RPC.

El método `trigger()` sigue estos pasos:

1. Si hay datos vinculados a la propiedad `params`, el método ejecuta todas las vinculaciones para garantizar que haya datos actualizados disponibles. Esto también provoca una validación de datos.
2. Si los datos no son válidos y el valor de `suppressInvalidCalls` es `true`, la operación se interrumpe.
3. Si la operación continúa, se emite el evento `send`.
4. La llamada remota real se inicia a través del método de conexión indicado (por ejemplo, HTTP).

Ejemplo

En este ejemplo se recupera un archivo XML remoto mediante el componente XMLConnector al establecer la propiedad `direction` en `receive`. Arrastre un componente XMLConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

En este código se especifica la URL del archivo XML y se establece `multipleSimultaneousAllowed` en `false`. Activa la instancia de XMLConnector tres veces, lo que hace que el método `status` del detector de eventos muestre el código de error `CallAlreadyInProgress` dos veces en el panel Salida. El primer intento se envía correctamente desde Flash al servidor. Cuando la primera activación recibe correctamente un resultado, se difunde un evento `result` y el paquete XML recibido se muestra en el panel Salida.

XMLConnector.URL

Disponibilidad

Flash Player 6 (6.0.79.0).

Edición

Flash MX Professional 2004.

Sintaxis

componentInstance.URL

Descripción

Propiedad; URL que este componente utiliza al realizar operaciones HTTP. Esta URL puede ser una URL absoluta o relativa. La URL está sujeta a todas las protecciones de seguridad de Flash Player estándar (para más información sobre las protecciones de seguridad de Flash Player, consulte "Aspectos básicos de la seguridad" en *Aprendizaje de ActionScript 2.0 en Flash*).

Ejemplo

En este ejemplo se recupera un archivo XML remoto mediante el componente XMLConnector al establecer la propiedad `direction` en `receive`. Arrastre un componente XMLConnector a su biblioteca e introduzca el siguiente código en el fotograma 1 de la línea de tiempo:

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

En este código se especifica la URL del archivo XML y se establece `multipleSimultaneousAllowed` en `false`. Activa la instancia de XMLConnector tres veces, lo que hace que el método `status()` del detector de eventos muestre el código de error `CallAlreadyInProgress` dos veces en el panel Salida. El primer intento se envía correctamente desde Flash al servidor. Cuando la primera activación recibe correctamente un resultado, se difunde un evento `result` y el paquete XML recibido se muestra en el panel Salida.

CAPÍTULO 60

Clase XPathAPI

60

Nombre de clase de ActionScript mx.xpath.XPathAPI

La clase XPathAPI permite realizar búsquedas XPath sencillas en Macromedia Flash. Puede resultar muy útil para buscar paquetes XML basados en nombres de nodo y valores de atributo. Dicho de otro modo, puede buscar rápidamente nodos y atributos en un documento XML mediante los métodos de XpathAPI.

Para utilizar las búsquedas XPath en Flash, primero debe incluir la clase XPathAPI en la biblioteca de Flash; para ello deberá añadir DataBindingClass (si no se ha añadido ya). Si ya ha configurado las vinculaciones, es posible que esta clase se haya incluido automáticamente. Si no es así, deberá seleccionar la clase en las bibliotecas comunes (Ventana > Bibliotecas comunes > Clases). En el panel de la biblioteca Classes.fla, simplemente arrastre una copia del componente DataBindingClasses en la biblioteca del documento de Flash actual. Después de esto podrá importar la clase si escribe `import mx.xpath.XPathAPI` o si utiliza el nombre completo de las clases cuando acceda a estos métodos, añadiendo el prefijo `mx.xpath.XPathAPI.`*`method_name`* a los métodos de la clase.

Para más información sobre esta clase, consulte el Centro de recursos de documentación de Flash en www.macromedia.com/go/xpathapi.

CAPÍTULO 61

Componente XUpdateResolver (sólo en Flash Professional)

61

Los componentes Resolver se utilizan con el componente DataSet (parte de la funcionalidad de administración de datos de la arquitectura de datos de Flash) para guardar los cambios en un origen de datos externo. Los componentes Resolver toman un objeto `DataSet.deltaPacket` y lo convierten en un paquete de actualización con un formato apropiado para el tipo de resolver. Los componentes Connector pueden transmitir a continuación el paquete de actualización al origen de datos externo. Los componentes Resolver no aparecen en pantalla en tiempo de ejecución.

Para obtener información general sobre cómo administrar datos en Flash mediante el componente DataSet, consulte "Administración de datos (sólo para Flash Professional)" en *Utilización de Flash*.

XUpdate es un estándar para describir los cambios que se realizan en un documento XML, y es compatible con diversas bases de datos XML, como Xindice y XHive. El componente XUpdateResolver convierte los cambios realizados a un componente DataSet en sentencias XUpdate. Las actualizaciones del componente XUpdateResolver se envían en forma de paquete de datos XUpdate, que se comunica a la base de datos o al servidor a través de un objeto de conexión. El componente XUpdateResolver obtiene un paquete delta de un componente DataSet, envía su propio paquete de actualización a un conector, recibe de la conexión errores de servidor y los transfiere de nuevo al componente DataSet. Todo esto mediante propiedades vinculables.

Para más información sobre el borrador de trabajo de la especificación de lenguaje XUpdate, consulte http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html. Para más información sobre la arquitectura de datos de Flash, consulte "Resolución de datos (sólo para Flash Professional)" en *Utilización de Flash*; para más información sobre la resolución de datos XML, consulte "Resolución de datos XML con el componente XUpdateResolver (sólo para Flash Professional)" en *Utilización de Flash*.

NOTA

También puede utilizar el componente XUpdateResolver para enviar actualizaciones de datos a un origen de datos externo que pueda analizar el lenguaje XUpdate (por ejemplo, una página ASP, un servlet Java o un componente de ColdFusion).

Utilización del componente XUpdateResolver (sólo en Flash Professional)

Utilice el componente XUpdateResolver sólo cuando la aplicación Flash contenga un componente DataSet y deba enviar de nuevo una actualización a un origen de datos externo.

El componente XUpdateResolver se comunica con el componente DataSet utilizando el codificador DataSetDeltaToXUpdateDelta. Este codificador crea sentencias XPath que identifiquen de forma exclusiva nodos en un archivo XML de acuerdo con la información del paquete delta del componente DataSet. Esta información la utiliza el componente XUpdateResolver para generar sentencias XUpdate. Para más información sobre el codificador DataSetDeltaToXUpdateDelta, consulte "Codificadores de esquema" en *Utilización de Flash*.

Para más información sobre el trabajo con el componente XUpdateResolver, consulte "Resolución de datos (sólo para Flash Professional)" en *Utilización de Flash*.

Parámetro del componente XUpdateResolver

El componente XUpdateResolver tiene un parámetro de edición, el parámetro booleano `includeDeltaPacketInfo`. Cuando se establece este parámetro en `true`, el paquete de actualización incluye información adicional que un origen de datos externo puede utilizar para generar resultados que pueden devolverse a la aplicación. Esta información incluye un ID exclusivo de transacción y operación que el juego de datos utiliza internamente.

NOTA	La información adicional que incluye el paquete de actualización invalida el XUpdate. Sólo se elegiría añadir esta información si se deseara almacenarla en un objeto de servidor y utilizarla para generar un paquete de resultados. En este caso, el objeto de servidor extraería la información necesaria del paquete de actualización y entonces pasaría el XUpdate (ahora válido) a la base de datos.

A continuación se muestra un ejemplo de un paquete de actualización XML cuando el parámetro `includeDeltaPacketInfo` se establece en `false`:

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/
  xupdate">
    <xupdate:remove select="/datapacket/row[@id='100']"/>
</xupdate:modifications>
```

A continuación se muestra un ejemplo de un paquete de actualización XML cuando el parámetro `includeDeltaPacketInfo` se establece en `true`:

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/
  xupdate"
      transId="46386292065:Wed Jun 25 15:52:34 GMT-0700 2003">
    <xupdate:remove select="/datapacket/row[@id='100']" opId="0123456789"/>
</xupdate:modifications>
```

Flujo de trabajo normal del componente XUpdateResolver

En el siguiente procedimiento se muestra el flujo de trabajo típico del componente XUpdateResolver.

Para utilizar un componente XUpdateResolver:

1. Añada a su aplicación dos instancias del componente XMLConnector y una instancia de cada uno de los componentes DataSet y XUpdateResolver y asígneles nombres de instancia.
2. Seleccione el primer componente XMLConnector y utilice la ficha Parámetros del inspector de componentes para introducir la URL del origen de datos XML externo al que desea acceder.
3. Con el componente XMLConnector seleccionado, haga clic en la ficha Esquema del inspector de componentes e importe un archivo XML de muestra para generar su esquema.

> **NOTA** Puede ser que necesite crear un esquema virtual para su archivo XML si desea acceder a un subelemento de la matriz a la que está vinculando el conjunto de datos. Para más información, consulte "Esquemas virtuales" en *Utilización de Flash*.

4. Utilice la ficha Vinculaciones del inspector de componentes para vincular una matriz del componente XMLConnector con la propiedad `dataProvider` del componente DataSet.
5. Seleccione el componente DataSet y utilice la ficha Esquema del inspector de componentes para crear los campos DataSet que se vincularán a los campos del objeto de la matriz.
6. Utilice la ficha Vinculaciones del inspector de componentes para vincular elementos de datos (campos DataSet) a los componentes visuales de su aplicación.
7. Seleccione la ficha Esquema del componente XUpdateResolver. Con la propiedad de componente `deltaPacket` seleccionada, utilice el panel Atributos de esquema para establecer la propiedad `encoder` para el codificador DataSetDeltaToXUpdateDelta.

8. Seleccione Opciones de codificador e introduzca el valor `rowNodeKey`, que identifica de manera exclusiva el nodo de fila en el archivo XML.

NOTA	El valor `rowNodeKey` combina una sentencia XPath con un parámetro de campo para definir la exclusividad con la cual deberían generarse las sentencias XPath para los datos de actualización que contiene el paquete delta. Para más información sobre el codificador DataSetDeltaToXUpdateDelta, consulte "Codificadores de esquema" en *Utilización de Flash*.

9. Haga clic en la ficha Vinculaciones y cree una vinculación entre la propiedad `deltaPacket` del componente XUpdateResolver y la propiedad `deltaPacket` del componente DataSet.
10. Cree otra vinculación desde la propiedad `xupdatePacket` al segundo componente XMLConnector para devolver los datos al origen de datos externo.

NOTA	La propiedad `xupdatePacket` contiene el paquete delta (sentencia XUpdate) con formato que se enviará al servidor.

11. Utilice un desencadenante para iniciar la operación de vinculación de datos: utilice el comportamiento Trigger Data Source asociado a un botón o añada ActionScript.

además de estos pasos, también puede crear vinculaciones para aplicar el paquete de resultados devuelto por el servidor al juego de datos mediante el componente XUpdateResolver.

Para ver un ejemplo paso a paso de la resolución de datos en un origen de datos externo mediante XUpdate, consulte "Update the timesheet" en los tutoriales de integración de datos en www.macromedia.com/go/data_integration.

Clase XUpdateResolver (sólo en Flash Professional)

Herencia MovieClip > XUpdateResolver

Nombre de clase de ActionScript mx.data.components.XUpdateResolver

Las propiedades y eventos de la clase XUpdateResolver permiten trabajar con el componente DataSet para guardar cambios en orígenes de datos externos.

Resumen de propiedades de la clase XUpdateResolver

En la tabla siguiente se enumeran las propiedades de la clase XUpdateResolver.

Propiedad	Descripción
`XUpdateResolver.deltaPacket`	Contiene una descripción de los cambios realizados en el componente DataSet. La propiedad `deltaPacket` del componente DataSet debería vincularse a esta propiedad para que, cuando se llame al método `DataSet.applyUpdates()`, se copie a través de la vinculación y el componente Resolver cree el paquete XUpdate.
`XUpdateResolver.includeDeltaPacketInfo`	Incluye información adicional del paquete delta en los atributos de los nodos XUpdate.
`XUpdateResolver.updateResults`	Describe los resultados de una actualización.
`XUpdateResolver.xupdatePacket`	Contiene la conversión XUpdate de los cambios realizados en el componente DataSet.

Resumen de eventos de la clase XUpdateResolver

En la tabla siguiente se enumeran los eventos de la clase XUpdateResolver.

Evento	Descripción
`XUpdateResolver.beforeApplyUpdates`	Lo llama el componente Resolver para realizar modificaciones personalizadas inmediatamente después de haber creado el paquete XML e inmediatamente antes de enviar dicho paquete.
`XUpdateResolver.reconcileResults`	Lo llama el componente Resolver para comparar dos paquetes.

XUpdateResolver.beforeApplyUpdates

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.beforeApplyUpdates(*eventObject*)

Parámetros

eventObject Objeto de evento Resolver; describe las personalizaciones del paquete XML antes de que la actualización se envíe a través del conector a la base de datos. Este objeto de evento debe contener las propiedades siguientes:

Propiedad	Descripción
target	Objeto; Resolver que genera este evento.
type	Cadena; nombre del evento.
updatePacket	Objeto XML; objeto XML que está a punto de aplicarse.

Valor devuelto

Ninguno.

Descripción

Evento; lo llama el componente Resolver para realizar modificaciones personalizadas inmediatamente después de haber creado el paquete XML para un nuevo paquete delta e inmediatamente antes de enviar dicho paquete mediante la vinculación de datos. Puede utilizar este controlador de eventos para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

Ejemplo

En el ejemplo siguiente se añaden los datos de autenticación de usuario al paquete XML:

```
on (beforeApplyUpdates) {
   // Añadir datos de autenticación de usuarios.
   var userInfo = new XML(""+getUserId()+" "+getPassword()+"");
   xupdatePacket.firstChild.appendChild(userInfo);
}
```

XUpdateResolver.deltaPacket

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.deltaPacket

Descripción

Propiedad; contiene una descripción de los cambios realizados en el componente DataSet. Esta propiedad es de tipo `deltaPacket`, recibe un paquete delta que debe convertirse en un paquete XUpdate y emite un paquete delta a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`. Esta propiedad proporciona un método para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade nuevamente un delta con mensajes al paquete delta para que se pueda reenviar la próxima vez que el paquete delta se envíe al servidor. Debe escribir código que gestione deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente paquete delta.

La propiedad `deltaPacket` del componente DataSet debería vincularse a esta propiedad para que, cuando se llame al método `DataSet.applyUpdates()`, se copie a través de la vinculación y el componente Resolver cree el paquete XUpdate.

XUpdateResolver.includeDeltaPacketInfo

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.includeDeltaPacketInfo

Descripción

Propiedad; propiedad booleana que, si su valor es `true`, incluye información adicional del paquete delta en los atributos de los nodos XUpdate. Esta información incluye los ID de transacción y de operación.

Para un ejemplo de XML resultante, consulte "Parámetro del componente XUpdateResolver" en la página 1550.

XUpdateResolver.reconcileResults

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

`resolveData``.reconcileResults(`*`eventObject`*`)`

Parámetros

`eventObject` Objeto ResolverEvent; describe el objeto de evento que se utiliza para comparar dos paquetes de actualización. Este objeto de evento debe contener las propiedades siguientes:

Propiedad	Descripción
`target`	Objeto; Resolver que genera este evento.
`type`	Cadena; nombre del evento.

Valor devuelto

Ninguno.

Descripción

Evento; lo llama el componente Resolver para comparar dos paquetes. Utilice esta función callback para insertar código después de que se hayan recibido los resultados del servidor e inmediatamente antes de la transmisión, a través de una vinculación de datos, del paquete delta que contiene los resultados de la operación. Es una buena ubicación para incluir código que gestione mensajes del servidor.

Ejemplo

En el ejemplo siguiente se igualan dos paquetes de actualización y se borran las actualizaciones si han sido correctas:

```
on (reconcileResults) {
      // Examinar resultados.
      if(examine(updateResults))
        myDataSet.purgeUpdates();
      else
       displayErrors(results);
}
```

XUpdateResolver.updateResults

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.updateResults

Descripción

Propiedad; propiedad del tipo `deltaPacket` que contiene los resultados de una actualización devuelta por el servidor mediante un conector. Utilice esta propiedad para transmitir errores y datos actualizados del servidor a un componente DataSet; por ejemplo, cuando el servidor asigna nuevos ID para un campo asignado automáticamente. Vincule esta propiedad con la propiedad `results` de un conector de manera que pueda recibir los resultados de una actualización y transmitirlos de nuevo al componente DataSet.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade nuevamente un delta con mensajes al paquete delta para que se pueda reenviar la próxima vez que el paquete delta se envíe al servidor. Debe escribir código que gestione deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente paquete delta.

XUpdateResolver.xupdatePacket

Disponibilidad

Flash Player 7.

Edición

Flash MX Professional 2004.

Utilización

resolveData.xupdatePacket

Descripción

Propiedad; propiedad de tipo `xml` que contiene la conversión XUpdate de los cambios realizados en el componente DataSet. Vincule esta propiedad a la propiedad del componente conector que transmite el paquete de actualización convertido de cambios de nuevo al componente DataSet.

Índice alfabético

A

B

C

D

E

F

I

J

L

M

N

O

P

R

S

T

U

V

W

X