



macromedia

FLASH

Utilización de componentes

8

Marcas comerciales

1 Step RoboPDF, ActiveEdit, ActiveTest, Authorware, Blue Sky Software, Blue Sky, Breeze, Breezo, Captivate, Central, ColdFusion, Contribute, Database Explorer, Director, Dreamweaver, Fireworks, Flash, FlashCast, FlashHelp, Flash Lite, FlashPaper, Flash Video Encoder, Flex, Flex Builder, Fontographer, FreeHand, Generator, HomeSite, JRun, MacRecorder, Macromedia, MXML, RoboEngine, RoboHelp, RoboInfo, RoboPDF, Roundtrip, Roundtrip HTML, Shockwave, SoundEdit, Studio MX, UltraDev y WebHelp son marcas registradas o marcas comerciales de Macromedia, Inc. y pueden estar registradas en Estados Unidos o en otras jurisdicciones, incluidas las internacionales. Otros nombres de productos, logotipos, diseños, títulos, palabras o frases mencionados en esta publicación pueden ser marcas comerciales, marcas de servicio o nombres comerciales de Macromedia, Inc. o de otras entidades y pueden estar registrados en ciertas jurisdicciones, incluidas las internacionales.

Información de terceros

Esta guía contiene vínculos a sitios Web de terceros que no están bajo el control de Macromedia y, por consiguiente, Macromedia no se hace responsable del contenido de dichos sitios Web. El acceso a uno de los sitios Web de terceros mencionados en esta guía será a cuenta y riesgo del usuario. Macromedia proporciona estos vínculos únicamente como ayuda y su inclusión no implica que Macromedia se haga responsable del contenido de dichos sitios Web.

La tecnología de compresión y descompresión de voz tiene licencia de Nellymoser, Inc. (www.nellymoser.com).



La tecnología de compresión y descompresión de vídeo Sorenson™ Spark™ tiene licencia de Sorenson Media, Inc.

Navegador Opera ® Copyright © 1995-2002 Opera Software ASA y sus proveedores. Todos los derechos reservados.

Macromedia Flash 8 utiliza tecnología de vídeo de On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Todos los derechos reservados. <http://www.on2.com>.

Visual SourceSafe es una marca registrada o un marca comercial de Microsoft Corporation en Estados Unidos y/u otros países.

Copyright © 2005 Macromedia, Inc. Todos los derechos reservados. No se permite la copia, fotocopia, reproducción, traducción ni la conversión en formato electrónico o legible por equipos, ya sea de forma total o parcial de este manual, sin la autorización previa por escrito de Macromedia, Inc. No obstante, el propietario o usuario autorizado de una copia válida del software con la que se proporcionó este manual puede imprimir una copia del manual a partir de una versión electrónica del mismo, con el solo fin de aprender a usar dicho software, siempre que no se imprima, reproduzca, revenda o transmita ninguna parte de este manual para cualquier otro propósito, incluidos, sin limitación, fines comerciales, como la venta de copias de esta documentación o el suministro de servicios de soporte pagados.

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: Mary Ferguson, Mary Kraemer, Lisa Stanziano

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 Bleye, 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

Introducción	7
Destinatarios	8
Requisitos del sistema.....	8
Documentación.....	8
Convenciones tipográficas.....	9
Términos utilizados en este manual	9
Recursos adicionales.....	9
Capítulo 1: Los componentes11
Instalación de componentes	12
Ubicación de almacenamiento de los archivos de componentes	14
Modificación de los archivos de componentes	15
Ventajas de utilizar componentes.....	16
Categorías de componentes	17
Arquitectura de componentes versión 2	18
Funciones de componentes de la versión 2.....	19
Clips compilados y archivos SWC	21
Accesibilidad y componentes	22
Capítulo 2: Creación de una aplicación con componentes (sólo en Flash Professional)	23
Tutorial de Fix Your Mistake	23
Creación de la página principal	25
Vinculación de componentes de datos para mostrar ideas para regalos.....	31
Visualización de detalles de los regalos	36
Creación de la página de finalización de la compra	42
Prueba de la aplicación.....	52
Visualización de la aplicación finalizada	52

Capítulo 3: Trabajo con componentes	53
Panel Componentes	54
Adición de componentes a documentos de Flash	54
Componentes del panel Biblioteca	58
Definición de parámetros de componentes	59
Ajuste del tamaño de los componentes	61
Eliminación de componentes de documentos de Flash	62
Utilización de las sugerencias para el código	62
Creación de un desplazamiento personalizado de la selección	63
Administración de la profundidad del componente en un documento	64
Componentes en previsualización dinámica	65
Utilización de un precargador con componentes	65
Carga de componentes	67
Actualización de componentes de la versión 1 de la arquitectura de componentes a la versión 2	68
Capítulo 4: Gestión de eventos de componentes	69
Utilización de detectores para gestionar eventos	70
Delegación de eventos	78
El objeto de evento	82
Utilización del controlador de eventos on()	83
Capítulo 5: Personalización de componentes	85
Utilización de estilos para personalizar el texto y el color de un componente	86
Aplicación de aspectos a los componentes	102
Temas	115
Combinación de aplicación de aspectos y estilos para personalizar un componente	126
Capítulo 6: Creación de componentes	133
Archivos de origen de componentes	133
Información general de la estructura de componentes	134
Creación del primer componente	136
Selección de una clase principal	146
Creación de un clip de película del componente	149
Creación del archivo de clase de ActionScript	154
Incorporación de componentes existentes en el componente	185
Exportación y distribución de un componente	195
Últimos pasos del desarrollo de componentes	199

Capítulo 7: Propiedades de colección	203
Definición de una propiedad de colección	205
Ejemplo sencillo de colección	206
Definición de la clase de un elemento de la colección	208
Acceso a información de colección mediante programación.	209
Exportación de componentes con colecciones a archivos SWC ...	211
Utilización de un componente que incluye una propiedad de colección	212
Índice alfabético	213

Introducción

Macromedia Flash Basic 8 y Macromedia Flash Professional 8 son las herramientas estándar de edición para la creación de publicaciones Web de gran impacto. Los componentes son bloques de creación para aplicaciones complejas de Internet que proporcionan dicho impacto. Un *componente* es un clip de película con parámetros que se establecen durante la edición en Macromedia Flash y con métodos, propiedades y eventos de ActionScript que permiten personalizar el componente en tiempo de ejecución. El diseño de los componentes permite a los desarrolladores volver a utilizar y compartir código, así como encapsular complejas funciones que los diseñadores pueden utilizar y personalizar sin necesidad de utilizar ActionScript.

Los componentes se basan en la versión 2 de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. En este manual se explica la forma de crear aplicaciones con componentes de la versión 2. En la *Referencia del lenguaje de componentes* relacionada, se describe la API (interfaz de programación de aplicaciones) de cada uno de los componentes. Se incluyen ejemplos de escenarios y procedimientos para utilizar con los componentes de la versión 2 de Flash, así como descripciones de las interfaces API de los componentes, en orden alfabético.

Puede utilizar los componentes creados por Macromedia, descargar componentes creados por otros desarrolladores o crear los suyos propios.

Este capítulo contiene las siguientes secciones:

Destinatarios	8
Requisitos del sistema	8
Documentación	8
Convenciones tipográficas	9
Términos utilizados en este manual	9
Recursos adicionales	9

Destinatarios

Este manual está dirigido a desarrolladores que crean aplicaciones de Flash y desean utilizar componentes para agilizar el desarrollo. Ya debe estar familiarizado con el desarrollo de aplicaciones en Flash y con la programación en ActionScript.

Si tiene menos experiencia programando en ActionScript, puede añadir componentes a un documento, establecer sus parámetros en el inspector de propiedades o el inspector de componentes, y usar el panel Comportamientos para gestionar sus eventos. Por ejemplo, puede asociar un comportamiento Ir a página Web a un componente Button que abre una URL en un navegador Web cuando se hace clic en el botón sin tener que escribir código ActionScript.

Si es un programador que desea crear aplicaciones más sólidas, puede crear componentes de forma dinámica, utilizar ActionScript para establecer propiedades y llamar a métodos en tiempo de ejecución, así como utilizar el modelo de detector de eventos para gestionar los eventos.

Para más información, consulte el [Capítulo 3, “Trabajo con componentes”](#), en la [página 53](#).

Requisitos del sistema

Los componentes de Macromedia no tienen ningún requisito del sistema adicional a los de Flash.

Cualquier archivo SWF que utilice componentes de la versión 2 debe verse con Flash Player 6 (6.0.79.0) o posterior, y debe publicarse para ActionScript 2.0 (puede definir este valor en Archivo > Configuración de publicación, en la ficha Flash).

Documentación

En este documento se explican los detalles de la utilización de componentes para desarrollar aplicaciones Flash. Se supone que tiene un conocimiento general de Macromedia Flash y ActionScript. La documentación específica sobre Flash y los productos relacionados se proporcionan por separado.

Este documento está disponible en formato de archivo PDF y como Ayuda en línea. Para ver la Ayuda en línea, inicie Flash y seleccione Ayuda > Utilización de componentes.

Para más información sobre Macromedia Flash, consulte los siguientes documentos:

- *Utilización de Flash*
- *Aprendizaje de ActionScript 2.0 en Flash*

- *Referencia del lenguaje ActionScript 2.0*
- *Referencia del lenguaje de componentes*

Convenciones tipográficas

En este manual se utilizan las siguientes convenciones tipográficas:

- La *f fuente en cursiva* indica un valor que se debe sustituir (por ejemplo, en una ruta de carpeta).
- La *f fuente para código* indica que se trata de código ActionScript, incluidos los nombres de métodos y propiedades.
- La *f fuente para código en cursiva* indica un elemento de código que se debe sustituir (por ejemplo, un parámetro de ActionScript).
- La *f fuente en negrita* indica un valor introducido por el usuario.

Términos utilizados en este manual

En este manual se utilizan los términos siguientes:

en tiempo de ejecución Cuando el código se ejecuta en Flash Player.

durante la edición Mientras se trabaja en el entorno de edición de Flash.

Recursos adicionales

Para obtener información reciente sobre Flash, además de las sugerencias de usuarios expertos, temas avanzados, ejemplos, consejos y otras actualizaciones, visite el sitio Web www.macromedia.com/es/devnet, que se actualiza con regularidad. Visite el sitio Web regularmente para conocer las últimas noticias sobre Flash y cómo sacar el máximo partido del programa.

Para más información sobre las notas técnicas, actualizaciones de documentación y vínculos a otros recursos de la comunidad de Flash, visite el centro de soporte de Macromedia Flash en www.macromedia.com/go/flash_support_es.

Para ver información detallada acerca de los términos, la sintaxis y el uso de ActionScript, consulte *Aprendizaje de ActionScript 2.0 en Flash* y *Referencia del lenguaje ActionScript 2.0*.

Para ver una introducción al uso de los componentes, consulte el seminario Macromedia On Demand titulado Using UI Components en www.macromedia.com/macromedia/events/online/ondemand/index.html.

Los componentes de Macromedia Flash son clips de película con parámetros que permiten modificar su aspecto y comportamiento. Pueden ser desde un sencillo control de interfaz de usuario como un botón de opción o una casilla de verificación, hasta un elemento con contenido, como un panel de desplazamiento; por otra parte, un componente también puede ser un elemento que no se ve, como FocusManager, que permite controlar qué objeto pasa a estar seleccionado en una aplicación.

Los componentes permiten a cualquier usuario crear complejas aplicaciones de Macromedia Flash, aunque no tenga conocimientos avanzados de ActionScript. En lugar de crear botones personalizados, cuadros combinados y listas, basta con arrastrar dichos componentes desde el panel Componentes para añadir funcionalidad a las aplicaciones. Asimismo, la apariencia de los componentes se puede personalizar fácilmente y conseguir, de esta manera, una mayor adaptación a las necesidades de diseño propias.

Los componentes se basan en la versión 2 de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. La arquitectura de la versión 2 incluye clases en las que se basan todos los componentes y los mecanismos de estilo y de aspecto que permiten personalizar la apariencia de los componentes, un modelo de eventos difusor/detector, administración de la profundidad y la selección, implementación de accesibilidad, etc.

NOTA

Cuando publique componentes de la versión 2, debe establecer la configuración de publicación en ActionScript 2.0 (Archivo > Configuración de publicación, ficha Flash). Los componentes de la versión 2 no se ejecutarán correctamente si se publican mediante ActionScript 1.0.

Cada componente dispone de parámetros predefinidos que se pueden establecer durante la edición en Flash. Asimismo, cada uno dispone de un conjunto único de métodos, propiedades y eventos de ActionScript, llamado también *API* (interfaz de programación de aplicaciones), que permite definir parámetros y opciones adicionales en tiempo de ejecución.

Para obtener una lista completa de los componentes que se incluyen con Flash Basic 8 y Flash Professional 8, consulte el “[Instalación de componentes](#)” en la [página 12](#). También puede descargar componentes creados por miembros de la comunidad de Flash desde el sitio Web de Macromedia Exchange, en la dirección www.macromedia.com/es/exchange/.

Este capítulo contiene las siguientes secciones:

Instalación de componentes	12
Ubicación de almacenamiento de los archivos de componentes	14
Modificación de los archivos de componentes	15
Ventajas de utilizar componentes	16
Categorías de componentes	17
Arquitectura de componentes versión 2	18
Funciones de componentes de la versión 2	19
Clips compilados y archivos SWC	21
Accesibilidad y componentes	22

Instalación de componentes

Al iniciar por primera vez Flash, ya hay un conjunto de componentes de Macromedia instalado. Pueden verse en el panel Componentes.

Flash Basic 8 incluye los componentes siguientes:

- [Componente Button](#)
- [Componente CheckBox](#)
- [Componente ComboBox](#)
- [Componente Label](#)
- [Componente List](#)
- [Componente Loader](#)
- [Componente NumericStepper](#)
- [Componente ProgressBar](#)
- [Componente RadioButton](#)
- [Componente ScrollPane](#)
- [Componente TextArea](#)
- [Componente TextInput](#)
- [Componente Window](#)

Flash Professional 8 incluye los componentes de Flash Basic 8, además de los siguientes componentes y clases adicionales:

- Componente Accordion (sólo en Flash Professional)
- Componente Alert (sólo en Flash Professional)
- Clases de vinculación de datos (sólo en Flash Professional)
- Componente DateField (sólo en Flash Professional)
- Componente DataGrid (sólo en Flash Professional)
- Componente DataHolder (sólo en Flash Professional)
- Componente DataSet (sólo en Flash Professional)
- Componente DateChooser (sólo en Flash Professional)
- Componente FLVPlayback (sólo en Flash Professional)
- Clase Form (sólo en Flash Professional)
- Componentes multimedia (sólo en Flash Professional)
- Componente Menu (sólo en Flash Professional)
- Componente MenuBar(sólo en Flash Professional)
- Componente RDBMSResolver (sólo en Flash Professional)
- Clase Screen (sólo en Flash Professional)
- Clase Slide (sólo en Flash Professional)
- Componente Tree (sólo en Flash Professional)
- Componente WebServiceConnector (sólo en Flash Professional)
- Componente XMLConnector (sólo en Flash Professional)
- Componente XUpdateResolver (sólo en Flash Professional)

Para ver los componentes de Flash Basic 8 o Flash Professional 8:

1. Inicie Flash.
2. Seleccione Ventana > Componentes para abrir el panel Componentes si aún no está abierto.
3. Seleccione User Interface para expandir el árbol y ver los componentes instalados.

También puede descargar componentes desde el sitio Web de Macromedia Exchange, en la dirección www.macromedia.com/go/exchange_es. Para instalar componentes descargados de Exchange, descargue e instale Macromedia Extension Manager desde www.macromedia.com/es/exchange/em_download/.

Cualquier componente puede aparecer en el panel Componentes de Flash. Realice estos pasos para instalar componentes en un equipo con Windows o Macintosh.

Para instalar componentes en un equipo con Windows o Macintosh:

1. Salga de Flash.
2. Coloque el archivo SWC o FLA que contiene el componente en la siguiente carpeta del disco duro:
 - En Windows: C:\Archivos de programa\Macromedia\Flex 8\idioma\Configuration\Components
 - En Macintosh: Disco duro de Macintosh/Applications/Macromedia Flash 8/Configuration/Components (Macintosh)
3. Inicie Flash.
4. Seleccione Ventana > Componentes para ver los componentes del panel Componentes, si aún no está abierto.

Ubicación de almacenamiento de los archivos de componentes

Los componentes de Flash se almacenan en la carpeta de configuración de nivel de aplicación.

NOTA

Para más información sobre estas carpetas, consulte “Carpetas de configuración instaladas con Flash” en *Primeros pasos con Flash*.

Los componentes se instalan en las siguientes ubicaciones:

- Windows 2000 o Windows XP: C:\Archivos de programa\Macromedia\Flex 8\idioma\Configuration\Components
- Mac OS X: Disco duro de Macintosh/Applications/Macromedia Flash 8/Configuration/Components

Modificación de los archivos de componentes

Los archivos de código fuente ActionScript para componentes se encuentran en:

- Windows 2000 o Windows XP: C:\Archivos de programa\Macromedia\Flash 8\idioma\First Run\Classes\mx
- Mac OS X: Disco duro de Macintosh/Applications/Macromedia Flash 8/First Run/Classes/mx

Los archivos del directorio First Run se copian en la ruta Documents and Settings cuando se inicia Flash por primera vez. Las rutas de Documents and Settings son:

- Windows 2000 o Windows XP: C:\Documents and Settings*usuario*\Local settings\Application Data\Macromedia\Flash 8\idioma\Configuration\Classes\mx
- Mac OS X: *Usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Classes/mx

Si se inicia Flash y falta un archivo de la ruta Document and Settings, Flash lo copia del directorio First Run a la ruta Documents and Settings.

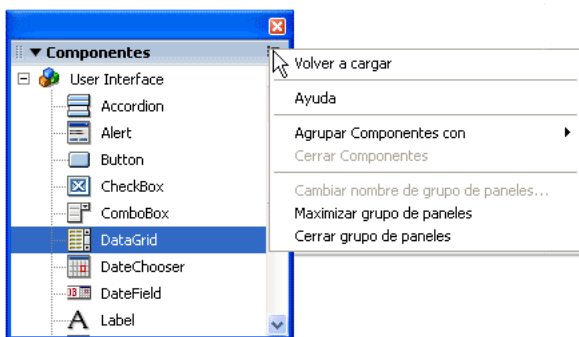
NOTA

Si desea modificar los archivos de código fuente ActionScript, modifíquelos en la ruta Documents and Settings. Si alguna de estas modificaciones produce errores en un componente, Flash copiará el archivo funcional del directorio First Run para restaurar la funcionalidad original al cerrar y reiniciar la aplicación. Sin embargo, si modifica los archivos en el directorio First Run y, como consecuencia, se producen errores en un componente, es posible que deba volver a instalar Flash para restaurar los archivos originales.

Si añade componentes, deberá actualizar el panel Componentes.

Para actualizar el contenido del panel Componentes:

- Seleccione Volver a cargar en el menú del panel Componentes.



Para eliminar un componente del panel Componentes:

- Elimine el archivo MXP o FLA de la carpeta de configuración.

Ventajas de utilizar componentes

Los componentes permiten separar el proceso de designación de la aplicación del proceso de programación. Asimismo, permiten reutilizar código, ya sea en componentes que se creen, o bien descargando e instalando componentes creados por otros desarrolladores.

Los componentes permiten a los programadores crear funciones que los diseñadores pueden utilizar en aplicaciones. Los desarrolladores pueden encapsular en componentes las funciones utilizadas con mayor frecuencia y los diseñadores pueden personalizar el aspecto y el comportamiento de los componentes cambiando los parámetros en el inspector de propiedades o en el inspector de componentes.

Los desarrolladores de Flash pueden intercambiar componentes a través del sitio Web de Macromedia Exchange, en la dirección www.macromedia.com/go/exchange_es. Si utiliza componentes, ya no tendrá que crear cada uno de los elementos de una aplicación Web compleja desde cero. Simplemente, busque los componentes que necesite y póngalos en un documento de Flash para crear una aplicación nueva.

Los componentes que se basan en la arquitectura de la versión 2 comparten una funcionalidad básica, como los estilos, la gestión de eventos, la aplicación de aspectos o la gestión de la selección y la profundidad. Al añadir el primer componente de la versión 2 a una aplicación, se añaden al documento alrededor de 25 KB con dicha funcionalidad básica. Si añade más componentes, esos mismos 25 KB se volverán a utilizar también para ellos, por lo que el tamaño del documento no aumentará tanto como parece. Para más información sobre cómo actualizar componentes, consulte [“Actualización de componentes de la versión 1 de la arquitectura de componentes a la versión 2”](#) en la página 68.

Categorías de componentes

Los componentes incluidos en Flash se clasifican en las cinco categorías siguientes (las ubicaciones de sus archivos de código fuente ActionScript también se corresponden aproximadamente con estas categorías y se muestran entre paréntesis):

- Componentes de datos (mx.data.*)

Los componentes de datos permiten cargar y manipular la información de los orígenes de datos; los componentes `WebServiceConnector` y `XMLConnector` son componentes de datos.

NOTA

Los archivos de origen de los componentes de datos no se instalan con Flash. No obstante, se instalan algunos de los archivos de ActionScript auxiliares.

- Componente `FLVPlayback` (mx.video.FLVPlayback)

El componente `FLVPlayback` 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).

- Componentes multimedia (mx.controls.*)

Los componentes multimedia permiten reproducir y controlar el flujo de medios. `MediaController`, `MediaPlayback` y `MediaDisplay` son componentes multimedia.

- Componentes de interfaz de usuario (mx.controls.*)

Los componentes de interfaz de usuario permiten interactuar con una aplicación. Por ejemplo, los componentes `RadioButton`, `CheckBox` y `TextInput` son controles de una interfaz de usuario.

- Administradores (mx.managers.*)

Los administradores son componentes que no se ven y que permiten administrar una función (por ejemplo, selección o profundidad) en una aplicación. `FocusManager`, `DepthManager`, `PopUpManager` y `StyleManager` son componentes administradores.

- Pantallas (mx.screens.*)

La categoría de pantallas incluye las clases de ActionScript que permiten controlar formularios y diapositivas en Flash.

Para obtener una lista completa de los componentes, consulte *Referencia del lenguaje de componentes*.

Arquitectura de componentes versión 2

Puede utilizar el inspector de propiedades o el inspector de componentes para cambiar parámetros de componente y utilizar las funciones básicas de los componentes. No obstante, si desea tener un mayor control sobre los componentes, tendrá que utilizar las interfaces API de los mismos y poseer conocimientos sobre cómo se crearon.

Los componentes de Flash se crean con la versión 2 de la arquitectura de componentes de Macromedia. Los componentes de la versión 2 son compatibles con Flash Player 6 (6.0.79.0) y posterior, y ActionScript 2.0. Dichos componentes no siempre son compatibles con los componentes creados mediante la versión 1 de la arquitectura (todos los componentes anteriores a Flash MX 2004). Además, los componentes originales de la versión 1 no son compatibles con Flash Player 7. Para más información, consulte “[Actualización de componentes de la versión 1 de la arquitectura de componentes a la versión 2](#)” en la página 68.

NOTA

Se han actualizado los componentes de interfaz de usuario de Flash MX para que funcionen con Flash Player 7 o posterior. Estos componentes actualizados aún se basan en la versión 1 de la arquitectura. Puede descargarlos desde el sitio Web de Macromedia Flash Exchange en www.macromedia.com/go/v1_components.

Los componentes de la versión 2 están incluidos en el panel Componentes como símbolos de clip compilado (SWC). Un clip compilado es un clip de película de componente cuyo código se ha compilado. Los clips compilados no pueden editarse, pero puede cambiar sus parámetros en el inspector de propiedades y en el inspector de componentes, de la misma manera que para cualquier componente. Para más información, consulte “[Clips compilados y archivos SWC](#)” en la página 21.

Los componentes de la versión 2 están programados en ActionScript 2.0. Cada componente es una clase y cada clase se encuentra en un paquete de ActionScript. Por ejemplo, un componente de botón de opción es una instancia de la clase `RadioButton` cuyo nombre de paquete es `mx.controls`. Para más información sobre paquetes, consulte “Paquetes” en *Aprendizaje de ActionScript 2.0 en Flash*.

La mayoría de los componentes de interfaz creados con la versión 2 de la arquitectura de componentes de Macromedia son subclases de las clases UIObject y UIComponent, y heredan todos los métodos, eventos y propiedades de dichas clases. Asimismo, muchos componentes son subclases de otros componentes. La ruta de herencia de cada componente se muestra en su entrada correspondiente en *Referencia del lenguaje de componentes*.

NOTA

La jerarquía de clases también está disponible como archivo de FlashPaper en la ubicación de instalación: Flash 8\Samples and Tutorials\Samples\Components\arch_diagram.swf.

Todos los componentes utilizan también el mismo modelo de eventos, los mismos estilos basados en CSS y los mismos temas y mecanismos de aplicación de aspectos incorporados. Para más información sobre los estilos y la aplicación de aspectos, consulte el [Capítulo 5, “Personalización de componentes”](#), en la [página 85](#). Para más información sobre la gestión de eventos, consulte el [Capítulo 3, “Trabajo con componentes”](#), en la [página 53](#).

Para ver una explicación detallada de la versión 2 de la arquitectura de componentes, consulte el [Capítulo 6, “Creación de componentes”](#), en la [página 133](#).

Funciones de componentes de la versión 2

En esta sección se describen las funciones de los componentes de la versión 2 (en comparación con los componentes de la versión 1) desde la perspectiva de un desarrollador que usa componentes para crear aplicaciones Flash. Para obtener información detallada sobre las diferencias entre la versión 1 y la versión 2 de la arquitectura para crear componentes, consulte el [Capítulo 6, “Creación de componentes”](#), en la [página 133](#).

El **inspector de componentes** permite cambiar los parámetros de los componentes al mismo tiempo que se edita en Macromedia Flash y en Macromedia Dreamweaver. (Véase [“Definición de parámetros de componentes”](#) en la [página 59](#).)

El **modelo de eventos de detector** permite a los detectores controlar eventos. (Véase [Capítulo 4, “Gestión de eventos de componentes”](#), en la [página 69](#).) Flash no incluye un parámetro `clickHandler` en el inspector de propiedades, como el que había en Flash MX; debe escribir código ActionScript para gestionar eventos.

Las **propiedades de aspecto** permiten cargar en tiempo de ejecución aspectos individuales, como flechas arriba y abajo, o la marca de una casilla de verificación. (Véase [“Aplicación de aspectos a los componentes”](#) en la [página 102](#).)

Los **estilos basados en CSS** permiten crear una apariencia uniforme en todas las aplicaciones. (Véase [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 86.](#))

Los **temas** permiten arrastrar una apariencia prediseñada desde la biblioteca a un conjunto de componentes. (Véase [“Temas” en la página 115.](#))

Halo es el tema que se usa de manera predeterminada en los componentes de la versión 2. (Véase [“Temas” en la página 115.](#))

Las **clases de administrador** proporcionan una manera fácil de gestionar la selección y la profundidad en una aplicación. (Véase [“Creación de un desplazamiento personalizado de la selección” en la página 63](#) y [“Administración de la profundidad del componente en un documento” en la página 64.](#))

Las **clases base UIObject y UIComponent** proporcionan métodos, propiedades y eventos básicos a los componentes que las amplían. (Consulte [“Clase UIComponent”](#) y [“Clase UIObject” en Referencia del lenguaje de componentes.](#))

El **empaquetado en archivos SWC** permite una fácil distribución y ocultación de código. Véase [Capítulo 6, “Creación de componentes”, en la página 133.](#)

La **vinculación de datos incorporados** está disponible a través del inspector de componentes. Para más información, consulte [“Integración de datos \(sólo para Flash Professional\)” en Utilización de Flash.](#)

Una **jerarquía de clases fácilmente ampliable** mediante ActionScript 2.0 permite crear espacios de nombres exclusivos, importar clases cuando sea necesario y derivar fácilmente subclases para ampliar componentes. Consulte [Capítulo 6, “Creación de componentes”, en la página 133](#) y [Referencia del lenguaje ActionScript 2.0.](#)

NOTA

Flash 8 incluye varias funciones no compatibles con los componentes de la versión 2 como, por ejemplo, la escala de 9 divisiones, FlashType y la caché de mapa de bits.

Clips compilados y archivos SWC

Un *clip compilado* es un paquete de código ActionScript y símbolos de Flash precompilados. Se usa para no tener que volver a compilar símbolos y código que no van a cambiar. Un clip de película también se puede “compilar” en Flash y convertir en un clip compilado. Por ejemplo, un clip de película con gran cantidad de código ActionScript que no cambie a menudo puede convertirse en un clip compilado. El clip compilado se comporta como el clip de película desde el que se compiló, aunque los clips compilados se muestran y se publican más rápido que los clips de película normales. Los clips compilados no se pueden editar, pero tienen propiedades que aparecen en el inspector de propiedades y en el inspector de componentes.

Los componentes incluidos con Flash no son archivos FLA. Son clips compilados que se han empaquetado en archivos SWC de clips compilados. SWC es el formato de archivo de Macromedia para distribuir componentes; contiene un clip compilado, el archivo de clase de ActionScript del componente y otros archivos que describen el componente. Para ver más detalles sobre los archivos SWC, consulte [“Exportación y distribución de un componente” en la página 195](#).

Si coloca un archivo SWC en la carpeta First Run/Components, el componente aparecerá en el panel Componentes. Si añade un componente al escenario desde el panel Componentes, se añadirá un símbolo de clip compilado a la biblioteca.

Para compilar un clip de película:

- Haga clic con el botón derecho del ratón (Windows) o presione Control y haga clic (Macintosh) en el clip de película en el panel Biblioteca y, a continuación, seleccione Convertir en clip compilado.

Para exportar un archivo SWC:

- Seleccione el clip de película en el panel Biblioteca, haga clic con el botón derecho del ratón (Windows) o presione la tecla Control y haga clic (Macintosh), y seleccione Exportar archivo SWC.

NOTA

Flash Basic 8 y Flash Professional 8 son compatibles con los componentes FLA.

Accesibilidad y componentes

Cada vez resulta más importante que el contenido Web sea accesible, es decir, que lo puedan utilizar usuarios con diferentes discapacidades. Los usuarios con deficiencias visuales pueden acceder al contenido visual de las aplicaciones Flash mediante la utilización de un software de lector de pantalla que proporciona una descripción hablada del contenido de la pantalla.

Cuando se crea un componente, el autor puede escribir código ActionScript que permite la comunicación entre el componente y el lector de pantalla. Cuando el desarrollador utiliza dicho componente para crear una aplicación en Flash, recurre al panel Accesibilidad para configurar la instancia de cada componente.

La mayoría de los componentes creados por Macromedia están diseñados teniendo en cuenta la accesibilidad. Para saber si un componente es accesible, consulte su entrada en *Referencia del lenguaje de componentes*. Cuando cree una aplicación en Flash, tendrá que añadir una línea de código por cada componente

```
(mx.accessibility.ComponentNameAccImpl.enableAccessibility();) y
```

definir los parámetros de accesibilidad en el panel Accesibilidad. La accesibilidad de los componentes funciona igual que para todos los clips de película de Flash.

Existe la posibilidad de navegar con el teclado por la mayoría de los componentes creados por Macromedia. En cada entrada de componente incluida en *Referencia del lenguaje de componentes* se indica si se puede controlar el componente mediante el teclado.

Los componentes de Flash son elementos creados previamente que se pueden utilizar al crear aplicaciones Macromedia Flash. Entre los componentes se incluyen los controles de interfaz de usuario, mecanismos de conectividad y acceso a datos y elementos multimedia. Los componentes permiten ahorrar trabajo al crear una aplicación Flash, pues proporcionan elementos y comportamientos que, de otra forma, deberían crearse partiendo de cero.

Este capítulo contiene un tutorial que muestra la forma de crear una aplicación Flash a partir de los componentes disponibles en Macromedia Flash Professional 8. Aprenderá a trabajar con componentes en el entorno de edición de Flash y a utilizar el código ActionScript para hacerlos interactivos.

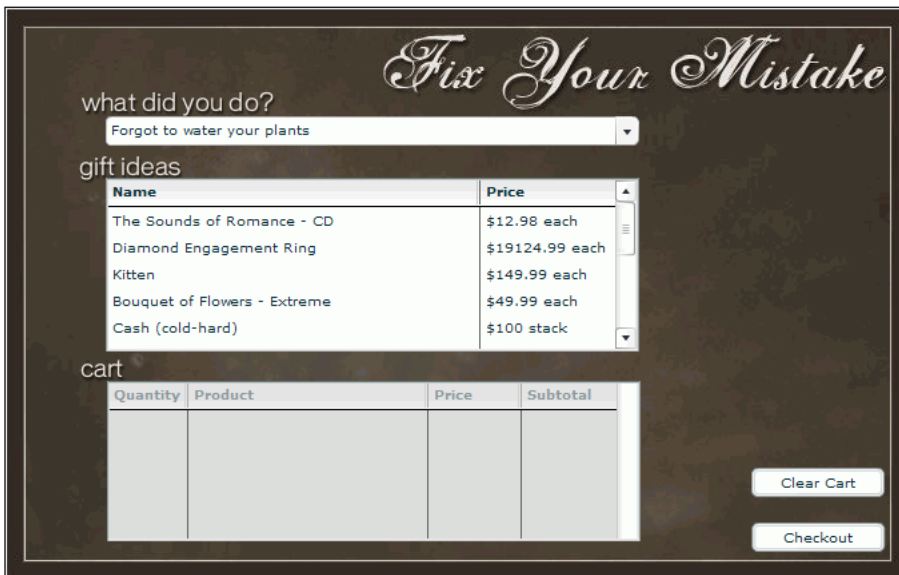
Tutorial de Fix Your Mistake

En este tutorial se describe el procedimiento para crear una aplicación básica de compra en línea para el servicio de regalos “Fix Your Mistake”. Este servicio ayuda a los usuarios a seleccionar un regalo adecuado si tuvieran que reconciliarse con alguien que se haya visto ofendido. La aplicación filtra una lista de regalos en función de la gravedad de la ofensa del usuario. El usuario puede añadir elementos de esa lista al carrito de la compra y pasar a la página de finalización de la compra para proporcionar los datos de facturación, envío y tarjeta de crédito.

Este capítulo contiene las siguientes secciones:

Tutorial de Fix Your Mistake	23
Creación de la página principal	25
Vinculación de componentes de datos para mostrar ideas para regalos.	31
Visualización de detalles de los regalos	36
Creación de la página de finalización de la compra	42
Prueba de la aplicación	52
Visualización de la aplicación finalizada	52

La aplicación utiliza los componentes ComboBox, DataGrid, TextArea y Button, entre otros, para crear la interfaz de la aplicación. La página principal de la interfaz tiene el siguiente aspecto:



La aplicación utiliza la clase WebService de ActionScript para conectar dinámicamente con un servicio y obtener la lista de ofensas (problems.xml) que aparecen en el cuadro combinado. También utiliza ActionScript para gestionar las interacciones del usuario con la aplicación.

La aplicación utiliza componentes de datos para conectar la interfaz a otro origen de datos. Utiliza el componente XMLConnector para conectarse a un archivo de datos XML (products.xml) y obtener la lista de regalos, y el componente DataSet para filtrar los datos y presentarlos en la cuadrícula de datos.

Para seguir el tutorial es necesario estar familiarizado con el entorno de edición de Flash y tener alguna experiencia con código ActionScript. Los usuarios deberán tener experiencia de uso de los elementos del entorno de edición, como los paneles, las herramientas, la línea de tiempo y la biblioteca. Con este tutorial se proporciona todo el código ActionScript necesario para crear la aplicación de ejemplo. No obstante, para comprender los conceptos sobre creación de scripts y crear sus propias aplicaciones, debe tener experiencia de programación en ActionScript.

Para ver una versión que funcione de la aplicación finalizada, consulte [“Visualización de la aplicación finalizada” en la página 52](#)

Recuerde que la aplicación de ejemplo sólo se ofrece con fines de demostración y que, por lo tanto, no es una aplicación finalizada real.

Creación de la página principal

Siga estos pasos para crear la página principal de la aplicación añadiendo componentes a una página inicial de estructura. A continuación, añada código ActionScript para personalizar los componentes, importe las clases de ActionScript que le permiten manipular los componentes de la aplicación y acceda a un servicio Web para llenar el cuadro combinado con una lista de ofensas. Para que el cuadro combinado se llene con el código, debe establecer su propiedad `dataProvider` para recibir los resultados del servicio Web.

1. Abra el archivo `first_app_start.fla`, que encontrará en una de las siguientes ubicaciones:
 - En Windows: *unidad de instalación*\Archivos de programa\Macromedia\Flesh 8\Samples and Tutorials\Samples\Components\ComponentsApplication
 - En Macintosh: *Disco duro de Macintosh*\Applications\Macromedia Flash 8\Samples and Tutorials\Samples\Components\ComponentsApplication

El archivo contiene una página inicial que tiene el siguiente aspecto:



El archivo `start_app.fla` contiene tres capas: una capa de fondo que contiene una imagen de fondo de color negro y títulos de texto, una capa de texto con etiquetas de texto para partes de la aplicación y una capa de etiquetas con etiquetas en el primer fotograma (Home) y en el décimo fotograma (Checkout).

2. Seleccione Archivo > Guardar como. Cambie el nombre del archivo y guárdelo en el disco duro.
3. En la línea de tiempo, seleccione la capa Labels y haga clic en el botón Insertar capa para añadir una nueva capa sobre ella. Asigne a la nueva capa el nombre **Form**. Colocará las instancias de componente en esta capa.
4. Asegúrese de que la capa Form está seleccionada. En el panel Componentes (Ventana > Componentes), busque el componente ComboBox en la carpeta User Interface. Arrastre una instancia de ComboBox al escenario. Colóquela bajo el texto What Did You Do? En el inspector de propiedades (Ventana > Propiedades > Propiedades), escriba el nombre de instancia **problems_cb**. Establezca la anchura en **400** píxeles. Introduzca el valor **76,0** para la ubicación *x* y el valor **82,0** para la ubicación *y*.

NOTA

El símbolo del componente ComboBox se añadirá a la biblioteca (Ventana > Biblioteca). Al arrastrar una instancia de un componente al escenario, se añade a la biblioteca el símbolo del clip compilado para el componente. Al igual que para todos los símbolos en Flash, puede crear instancias adicionales del componente arrastrando el símbolo desde la biblioteca al escenario.

5. Arrastre una instancia del componente DataGrid desde el árbol User Interface del panel Componentes al escenario. Colóquela bajo el texto Gift Ideas. Especifique **products_dg** como nombre de instancia. Establezca la anchura en **400** píxeles y la altura en **130** píxeles. Introduzca el valor **76,0** para la ubicación *x* y el valor **128,0** para la ubicación *y*.
6. Arrastre una instancia del componente DataSet desde el árbol Data del panel Componentes al escenario. (El componente DataSet no se ve en la aplicación en tiempo de ejecución. El icono DataSet es simplemente un marcador de posición con el que trabajará en el entorno de edición de Flash.) Especifique **products_ds** como nombre de instancia.

Arrastre una instancia del componente XMLConnector desde el árbol Data del panel Componentes al escenario. (al igual que el componente DataSet, el componente XMLConnector no se ve en la aplicación en tiempo de ejecución). Especifique **products_xmlcon** como nombre de instancia. Haga clic en la ficha Parámetros del inspector de propiedades y especifique www.flash-mx.com/mm/firstapp/products.xml como valor de la propiedad URL. Haga clic en el valor de la propiedad direction para activar el cuadro combinado y, a continuación, haga clic en la flecha abajo y seleccione *receive* en la lista.

NOTA

También puede utilizar el inspector de componentes (Ventana > Inspector de componentes) para configurar los parámetros de los componentes. En el inspector de propiedades y el inspector de componentes, la ficha Parámetros funciona de la misma manera.

La URL especifica un archivo XML externo con datos sobre los productos que aparecen en la sección Gift Ideas de la aplicación. En una fase posterior del tutorial usará la vinculación de datos para vincular entre sí los componentes XMLConnector, DataSet y DataGrid; el componente DataSet filtra datos del archivo XML externo y el componente DataGrid los muestra.

7. Arrastre una instancia del componente Button desde el árbol User Interface del panel Componentes al escenario. Colóquela en la esquina inferior derecha del escenario. Especifique **checkout_button** como nombre de instancia. Haga clic en la ficha Parámetros y escriba **Checkout** como valor de la propiedad label. En las coordenadas *x* e *y*, introduzca **560,3** y **386,0** respectivamente.

Importación de las clases de componente

Cada componente está asociado a un archivo de clase de ActionScript que define sus métodos y propiedades. En esta sección del tutorial, añadirá código ActionScript para importar las clases asociadas con los componentes de la aplicación. Para algunos de estos componentes ya ha añadido instancias al escenario. Para otros, añadirá código ActionScript en una fase posterior del tutorial para crear instancias dinámicamente.

La sentencia `import` crea una referencia al nombre de la clase y facilita la escritura de código ActionScript para el componente. La sentencia `import` permite hacer referencia a la clase a través de su nombre de clase, en lugar de hacerlo a través de su nombre completo, que incluye el nombre del paquete. Por ejemplo, cuando cree una referencia al archivo de clase `ComboBox` con una sentencia `import`, puede hacer referencia a instancias del cuadro combinado con la sintaxis `instanceName:ComboBox`, en lugar de `instanceName:mx.controls.ComboBox`.

Un *paquete* es un directorio que contiene archivos de clase y reside en un directorio de ruta de clases designado. Puede utilizar un carácter comodín para crear referencias a todas las clases de un paquete: por ejemplo, la sintaxis `mx.controls.*` crea referencias a todas las clases del paquete de controles. (Al crear una referencia a un paquete con un comodín, se quitan las clases no utilizadas de la aplicación cuando se compile, para que no aumenten el tamaño.)

En la aplicación referida en este tutorial se necesitan los siguientes paquetes y clases individuales:

Paquete UI Components Controls Este paquete contiene clases para los componentes de controles de la interfaz de usuario, incluidos ComboBox, DataGrid, Loader, TextInput, Label, NumericStepper, Button y CheckBox.

Paquete UI Components Containers Este paquete contiene clases para los componentes de contenedor de la interfaz de usuario, incluidos Accordion, ScrollPane y Window. Al igual que en el caso del paquete de controles, puede crear una referencia a este paquete mediante un comodín.

Clase DataGridColumn Esta clase permite añadir columnas a la instancia de DataGrid y controlar su apariencia.

Clase WebService Esta clase llena la instancia de ComboBox con una lista de problemas u ofensas. Para esta clase, también tendrá que importar el elemento WebServiceClasses de la biblioteca común Classes. Este elemento contiene archivos (SWC) de clips compilados que necesitará para compilar y generar el archivo SWF para la aplicación.

Clase Cart Una clase personalizada proporcionada con este tutorial, que define el funcionamiento del carrito de la compra que creará más adelante. (Para examinar el código del archivo de la clase Cart, abra el archivo `cart.as` que se encuentra en la carpeta `component_application` con los archivos FLA y SWF de la aplicación).

Para importar estas clases, creará una capa Actions y añadirá el código ActionScript al primer fotograma de la línea de tiempo principal. Todo el código que añada a la aplicación en los pasos restantes del tutorial se colocará en la capa Actions.

1. Para importar el elemento WebServiceClasses de la biblioteca Classes, seleccione Ventana > Bibliotecas comunes > Clases.
2. Arrastre el elemento WebServiceClasses desde la biblioteca Classes a la biblioteca para la aplicación.

Importar un elemento de la biblioteca Classes es similar a añadir un componente a la biblioteca: añade los archivos SWC de la clase a la biblioteca. Para poder usar la clase en una aplicación, los archivos SWC deben estar en la biblioteca.

3. En la línea de tiempo, seleccione la capa Form y haga clic en el botón Insertar capa. Asigne a la nueva capa el nombre **Actions**.

4. Con la capa Accions seleccionada, seleccione el fotograma 1 y presione F9 para abrir el panel Acciones.
5. En el panel Acciones, escriba el código siguiente para crear una función `stop()` que evite que la aplicación entre en un bucle infinito durante su reproducción:

```
stop();
```

6. Con el fotograma 1 en la capa Accions aún seleccionado, añada el código siguiente en el panel Acciones para importar las clases:

```
// Importar las clases necesarias.  
import mx.services.Webservice;  
import mx.controls.*;  
import mx.containers.*;  
import mx.controls.gridclasses.DataGridColumn;  
// Importar la clase Cart personalizada.  
import Cart;
```

Asignación de tipos de datos a las instancias de componentes

A continuación asignaré tipos de datos a cada una de las instancias de componentes que arrastré al escenario en las fases anteriores del tutorial.

ActionScript 2.0 usa tipos de datos estrictos, lo que significa que al crear una variable se debe asignar el tipo de datos. Al usar tipos de datos estrictos estarán disponibles sugerencias de código para la variable en el panel Acciones.

- En el panel Acciones, añada el código siguiente para asignar tipos de datos a las cuatro instancias de componente que ya creó.

```
/* Instancias de tipo de datos en el escenario; pueden añadirse otras  
instancias en  
tiempo de ejecución desde la clase Cart.*/  
var problems_cb:ComboBox;  
var products_dg:DataGrid;  
var cart_dg:DataGrid;  
var products_xmlcon:mx.data.components.XMLConnector;
```

NOTA

Los nombres de instancia que especifique aquí deben coincidir con los nombres de instancia que asignó al arrastrar los componentes al escenario.

Personalización de la apariencia de los componentes

Cada componente tiene propiedades y métodos de estilo (incluidos el color de resaltado, la fuente y el tamaño de fuente) que permiten personalizar su apariencia. Puede establecer estilos para instancias de componente individuales o establecer estilos globalmente para aplicarlos a todas las instancias de componentes de una aplicación. En este tutorial va a establecer estilos globalmente.

- Añada el código siguiente para establecer estilos:

```
// Definir estilos globales y ecuaciones de aceleración para la instancia
problems_cb de ComboBox.
_global.style.setStyle("themeColor", "haloBlue");
_global.style.setStyle("fontFamily", "Verdana");
_global.style.setStyle("fontSize", 10);
_global.style.setStyle("openEasing",
    mx.transitions.easing.Bounce.easeOut);
```

Este código establece el color del tema (el color de resaltado en un elemento seleccionado), la fuente y el tamaño de fuente para los componentes, y también establece el suavizado para el componente ComboBox (la forma en que aparece y desaparece la lista desplegable al hacer clic en la barra de título del componente ComboBox).

Visualización de ofensas en el cuadro combinado

En esta sección añadirá código para conectarse con un servicio Web que contiene la lista de ofensas (Forgot to Water Your Plants, etc.). El archivo de lenguaje de descripción de servicios Web (WSDL) se encuentra en www.flash-mx.com/mm/firstapp/problems.cfc?WSDL. Para ver cómo está estructurado el archivo WSDL, navegue hasta la ubicación de WSDL.

El código ActionScript pasa los resultados del servicio Web a la instancia de ComboBox para mostrarlos. Una función ordena las ofensas por orden de gravedad. Si el servicio Web no devuelve ningún resultado (por ejemplo, si el servicio no está activo o no se encuentra la función), aparece un mensaje de error en el panel Salida.

- En el panel Acciones, añada el código siguiente:

```
/* Definir el servicio Web usado para recuperar una matriz de problemas.
Este servicio se vinculará a la instancia problems_cb de ComboBox. */
var problemService:WebService = new WebService("http://www.flash-mx.com/
    mm/firstapp/problems.cfc?WSDL");
var myProblems:Object = problemService.getProblems();

/* Si se obtiene un resultado del servicio Web, establecer el campo que
se usará para la etiqueta de columna.
Establecer como proveedor de datos los resultados devueltos por el
servicio Web. */
myProblems.onResult = function(wsdResults:Array) {
```

```

        problems_cb.labelField = "name";
        problems_cb.dataProvider = wsdlResults.sortOn("severity",
        Array.NUMERIC);
    };

    /* Si no se puede conectar con el servicio Web remoto, mostrar los
    mensajes de error en el panel Salida. */
    myProblems.onFault = function(error:Object) {
        trace("error:");
        for (var prop in error) {
            trace("  "+prop+" -> "+error[prop]);
        }
    };
};

```

SUGERENCIA

Presione Control+S para guardar el trabajo y, a continuación, Control+Intro (o seleccione Control > Probar película) para probar la aplicación. En este momento, el cuadro combinado debería llenarse con una lista de ofensas y debería ver la cuadrícula de datos vacía que creó para la sección Gift Ideas, junto con el botón de finalización de la compra (Checkout).

Vinculación de componentes de datos para mostrar ideas para regalos

En el principio del tutorial añadió instancias de los componentes DataGrid, DataSet y XMLConnector al escenario. Estableció la propiedad URL para la instancia de XMLConnector denominada products_xmlcon en la ubicación de un archivo XML que contiene información de productos para la sección Gift Ideas de la aplicación.

Ahora usará las características de vinculación de datos del entorno de edición de Flash para vincular entre sí los componentes XMLConnector, DataSet y DataGrid con el fin de que usen los datos XML de la aplicación. Para ver información general sobre cómo trabajar con vinculación de datos y otras características de la arquitectura de integración de datos de Flash, consulte Capítulo 16, “Integración de datos (sólo para Flash Professional)” en *Utilización de Flash*.

Al vincular los componentes, el componente DataSet filtra la lista de productos del archivo XML en función de la gravedad de la ofensa que el usuario seleccione en la sección What Did You Do? El componente DataGrid mostrará la lista.

Utilización del esquema para describir el origen de datos XML

Al conectar con un origen de datos XML externo mediante el componente XMLConnector, debe especificar un *esquema* (una representación esquemática que describe la estructura del documento XML). El esquema indica al componente XMLConnector cómo debe leer el origen de datos XML. La forma más sencilla de especificar un esquema es importar una copia del archivo XML con el que se va a conectar y usar dicha copia como esquema.

1. Inicie el navegador Web y vaya a www.flash-mx.com/mm/firstapp/products.xml (la ubicación establecida para el parámetro URL de XMLConnector).
2. Seleccione Archivo > Guardar como.
3. Guarde products.xml en la misma ubicación que el archivo FLA en el que está trabajando.
4. Seleccione el fotograma 1 de la línea de tiempo.
5. Seleccione la instancia products_xmlcon (XMLConnector) junto al escenario.
6. En el inspector de componentes, haga clic en la ficha Esquema. Haga clic en el botón Importar (en el lado derecho de la ficha Esquema, sobre el panel de desplazamiento). En el cuadro de diálogo Abrir, busque el archivo products.xml que importó en el paso 3 y haga clic en Abrir. El esquema del archivo products.xml aparece en el panel de desplazamiento de la ficha Esquema.

En el panel superior de la ficha Esquema, seleccione el elemento `image`. En el panel inferior, seleccione `data type` y cambie el valor de `<vacío>` a `String`. Repita este paso para el elemento `description`.

Filtro de las ideas para regalos según la ofensa

Usará la ficha Vinculaciones del inspector de componentes para vincular entre sí las instancias de los componentes XMLConnector, DataSet y DataGrid.

Para más información sobre cómo trabajar con la vinculación de datos, consulte “Integración de datos (sólo para Flash Professional)” en *Utilización de Flash*.

1. Con la instancia products_xmlcon (XMLConnector) seleccionada en el escenario, haga clic en la ficha Vinculaciones del inspector de componentes.
2. Haga clic en el botón Añadir vinculación.
3. En el cuadro de diálogo Añadir vinculación, seleccione el elemento `results.products.product array` y haga clic en Aceptar.
4. En la ficha Vinculaciones, haga clic en el elemento Vinculado a del panel Atributos de vinculación (el panel inferior, en el que se muestran pares nombre-valor de atributos).

5. En la columna Valor del elemento Vinculado a, haga clic en el icono de lupa para abrir el cuadro de diálogo Vinculado a.
6. En el cuadro de diálogo Vinculado a, seleccione la instancia <products_ds> de DataSet en el panel Ruta del componente. Seleccione dataProvider:array en el panel Ubicación del esquema. Haga clic en Aceptar.
7. En la ficha Vinculaciones, haga clic en el elemento Dirección del panel Atributos de vinculación. En el menú desplegable de la columna Valor, seleccione Fuera.
Esta opción significa que los datos pasarán de la instancia products_xml con a la instancia products_ds (en lugar de pasar en ambos sentidos, o pasar de la instancia de DataSet a la instancia de XMLConnector).
8. En el escenario, seleccione la instancia products_ds. En la ficha Vinculaciones del inspector de componentes, observe que el proveedor de datos del componente aparece en la Lista de vinculaciones (el panel superior de la ficha Vinculaciones). En el panel Atributos de vinculación, el parámetro Vinculado a indica que la instancia products_ds está vinculada a la instancia products_xml con y que el sentido de vinculación es Dentro.
En los pasos siguientes vinculará la instancia de DataSet a la instancia de DataGrid de forma que los datos filtrados por el juego de datos se mostrarán en la cuadrícula de datos.
9. Con la instancia products_ds aún seleccionada, haga clic en el botón Añadir vinculación de la ficha Vinculaciones.
10. En el cuadro de diálogo Añadir vinculación, seleccione el elemento dataProvider: array y haga clic en Aceptar.
11. En la ficha Vinculaciones, asegúrese de que el elemento dataProvider: array está seleccionado en la Lista de vinculaciones.
12. Haga clic en el elemento Vinculado a del panel Atributos de vinculación.
13. En la columna Valor del elemento Vinculado a, haga clic en el icono de lupa para abrir el cuadro de diálogo Vinculado a.
14. En el cuadro de diálogo Vinculado a, seleccione la instancia products_dg (DataGrid) en el panel Ruta del componente. Seleccione dataProvider:array en el panel Ubicación del esquema. Haga clic en Aceptar.

Adición de columnas a la sección Gift Ideas

Ya está preparado para añadir columnas a la cuadrícula de datos en la sección Gift Ideas de la aplicación, a fin de mostrar información y precios de productos.

- Seleccione la capa Actions. En el panel Acciones, añada el código siguiente para crear, configurar y añadir una columna Name y una columna Price a la instancia de DataGrid:

```
// Definir columnas de cuadrícula de datos y sus anchuras predeterminadas
    en la
// instancia de DataGrid products_dg.
var name_dgc:DataGridColumn = new DataGridColumn("name");
name_dgc.headerText = "Name";
name_dgc.width = 280;

// Añadir la columna al componente DataGrid.
products_dg.addColumn(name_dgc);
var price_dgc:DataGridColumn = new DataGridColumn("price");
price_dgc.headerText = "Price";
price_dgc.width = 100;

// Definir la función que se usará para establecer las etiquetas de las
    columnas
// en tiempo de ejecución.
products_dgc.labelFunction = function(item:Object) {
    if (item != undefined) {
        return "$"+item.price+" "+item.priceQualifier;
    }
};
products_dg.addColumn(price_dgc);
```

Activación de XMLConnector

A continuación, añadirá una línea de código que hace que la instancia de XMLConnector cargue, analice y vincule el contenido del archivo products.xml remoto. Este archivo se encuentra en la URL que especificó para la propiedad URL de la instancia de XMLConnector que creó anteriormente. El archivo contiene información sobre los productos que se mostrarán en la sección Gift Ideas de la aplicación.

- Introduzca el código siguiente en el panel Acciones:

```
products_xmlcon.trigger();
```

Adición de un detector de eventos para filtrar ideas para regalos

En esta sección, añadirá un detector de eventos para detectar que un usuario selecciona una ofensa en la sección What Did You Do? (la instancia `problems_cb` de `ComboBox`). El detector incluye una función que filtra la lista de ideas para regalos según la ofensa que elija el usuario. Si selecciona una ofensa poco importante, se mostrará una lista de regalos modestos (como un CD o flores); si selecciona una ofensa más grave, se mostrarán regalos más caros.

Para más información sobre cómo trabajar con detectores de eventos, consulte “Utilización de detectores de eventos” en *Aprendizaje de ActionScript 2.0 en Flash*.

- En el panel Acciones, añada el código siguiente:

```
/* Definir un detector para la instancia problems_cb de ComboBox.
Este detector filtrará los productos en el componente DataSet (y
DataSet).
El filtrado se basa en la gravedad del elemento seleccionado actualmente
en el componente ComboBox. */
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
    products_ds.filtered = false;
    products_ds.filtered = true;
    products_ds.filterFunc = function(item:Object) {
        // Si la gravedad del elemento actual es mayor o igual que la del
        // elemento seleccionado en el componente ComboBox, devolver true.
        return (item.severity>=evt.target.selectedItem.severity);
    };
};

// Añadir el detector al componente ComboBox.
problems_cb.addEventListener("change", cbListener);
```

Si restablece el valor de la propiedad `filtered` (estableciéndolo en `false` y después en `true`) al principio de la función `change()`, se asegura de que la función se ejecutará correctamente en caso de que el usuario cambie la selección de la sección What Did You Do? repetidamente.

La función `filterFunc()` comprueba si un elemento determinado de la matriz de regalos está dentro del nivel de gravedad que el usuario seleccionó en el cuadro combinado. Si el regalo está dentro del rango de gravedad seleccionado, se muestra en la instancia de `DataSet` (que está vinculada a la instancia de `DataSet`).

La última línea de código registra el detector en la instancia `problems_cb` de `ComboBox`.

Adición del carrito

El código siguiente que va añadir crea una instancia de la clase `Cart` personalizada y después la inicializa.

- En el panel Acciones, añada el código siguiente:

```
var myCart:Cart = new Cart(this);  
myCart.init();
```

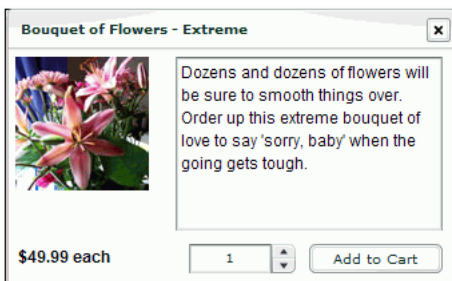
Este código usa el método `init()` de la clase `Cart` para añadir una instancia de `DataGrid` al escenario, define las columnas y establece la posición de la instancia en el escenario. También añade una instancia del componente `Button` y establece su posición, y añade un controlador `Alert` para el botón. (Para ver el código del método `init()` de la clase `Cart`, abra el archivo `Cart.as`.)

SUGERENCIA

Presione `Control+S` para guardar el trabajo y, a continuación `Control+Intro` (o seleccione `Control->Probar película`) para probar la aplicación. Cuando selecciona una ofensa en el cuadro combinado, la cuadrícula de datos que creó para las ideas para regalos debería mostrar un subconjunto de regalos correspondientes a la ofensa seleccionada.

Visualización de detalles de los regalos

En la aplicación aparece una ventana emergente cuando un usuario hace clic en un producto de la sección `Gift Ideas`. La ventana emergente contiene instancias de componente que muestran información del producto, incluido un texto descriptivo, una imagen y el precio. Para crear esta ventana emergente deberá crear un símbolo de clip de película y añadir instancias de los componentes `Loader`, `TextArea`, `Label`, `NumericStepper` y `Button`. La ventana de detalles del producto `Bouquet of Flowers Extreme` tiene el siguiente aspecto:



En una fase posterior añadirá código `ActionScript` que creará dinámicamente una instancia de este clip de película para cada producto. Estas instancias de clip de película se mostrarán en el componente `Window` que añadió anteriormente a la biblioteca. Las instancias de componentes se llenarán con elementos del archivo XML externo.

1. Arrastre una instancia del componente `Window` desde el árbol `User Interface` del panel Componentes al escenario.

Se añade el símbolo del componente `Window` a la biblioteca. En una fase posterior del tutorial creará instancias del componente `Window` mediante código `ActionScript`.

2. En el panel Biblioteca (Ventana > Biblioteca), haga clic en el menú de opciones del lado derecho de la barra de título y seleccione `Nuevo símbolo`.
3. En el cuadro de diálogo `Crear un nuevo símbolo`, escriba `ProductForm` como valor de `Nombre` y seleccione `Clip de película` como valor de `Tipo`.
4. Haga clic en el botón `Avanzado`. En `Vinculación`, seleccione `Exportar para ActionScript`, deje seleccionada la opción `Exportar en primer fotograma` y haga clic en `Aceptar`. Se abrirá una ventana de documento para el nuevo símbolo en modo de edición de símbolos.

Para símbolos de clip de película que están en la biblioteca pero no en el escenario, debe seleccionar `Exportar para ActionScript`, de forma que pueda manipularlos mediante código `ActionScript`. (`Exportar en el primer fotograma` significa que el clip de película estará disponible en cuanto se haya cargado el primer fotograma.) En una fase posterior del tutorial añadirá código `ActionScript` para generar una instancia del clip de película dinámicamente cada vez que un usuario haga clic en un producto de la sección `Gift Ideas`.

5. En la línea de tiempo para el nuevo símbolo, seleccione la capa 1 y asígnele el nombre `Components`.
6. Arrastre una instancia del componente `Loader` desde el árbol `User Interface` del panel Componentes al escenario. Escriba `5, 5` en las coordenadas `x, y` respectivamente. Especifique el nombre de instancia `image_1dr`. Haga clic en la ficha `Parámetros` del inspector de propiedades. Seleccione `false` para `autoLoad` y `false` para `scaleContent`. La instancia del componente `Loader` se usará para mostrar una imagen del producto. La configuración `false` de `autoLoad` especifica que la imagen no se cargará automáticamente. La configuración `false` de `scaleContent` especifica que no se ajustará la escala de la imagen. En una fase posterior del tutorial añadirá código para cargar la imagen dinámicamente, en función del producto que elija el usuario en la sección `Gift Ideas`.

7. Arrastre una instancia del componente `TextArea` desde el árbol `User Interface` del panel `Componentes` al escenario. Colóquela junto al componente `Loader`. Escriba `125, 5` en las coordenadas x, y respectivamente. Especifique el nombre de instancia **`description_ta`**. Establezca una `Anchura` de **`200`** y una `Altura` de **`130`**. Haga clic en la ficha `Parámetros` del inspector de propiedades. Para `editable`, seleccione `false`. Para `html`, seleccione `true`. Para `wordWrap`, seleccione `true`.

La instancia del componente `TextArea` se usa para mostrar un texto descriptivo del producto seleccionado. La configuración seleccionada especifica que el texto no puede ser editado por un usuario, se puede formatear con etiquetas HTML y que se ajustarán las líneas para que quepa en el área de texto.

8. Arrastre una instancia del componente `Label` desde el árbol `User Interface` del panel `Componentes` al escenario. Colóquela debajo del componente `Loader`. Establezca la coordenadas x, y en los valores `5, 145`. Especifique **`price_lbl`** como nombre de instancia. Haga clic en la ficha `Parámetros` del inspector de propiedades. Para `autoSize`, seleccione `left`. Para `html`, seleccione `true`.

La instancia del componente `Label` mostrará el precio del producto y el calificador del precio (la cantidad de productos indicados por el precio especificado, como “cada” o “una docena”).

9. Arrastre una instancia del componente `NumericStepper` desde el árbol `User Interface` del panel `Componentes` al escenario. Colóquela debajo del componente `TextArea`. Establezca la coordenadas x, y en los valores `135, 145`. Especifique el nombre de instancia **`quantity_ns`**. Haga clic en la ficha `Parámetros` del inspector de propiedades. Como valor de `minimum`, escriba `1`.

Si establece el valor de `minimum` en `1`, especifica que el usuario debe seleccionar al menos uno de los productos para añadir el elemento al carrito.

10. Arrastre una instancia del componente `Button` desde el árbol `User Interface` del panel `Componentes` al escenario. Colóquela junto al componente `NumericStepper`. Establezca la coordenadas x, y en los valores `225, 145`. Especifique **`addToCart_button`** como nombre de instancia. Haga clic en la ficha `Parámetros` del inspector de propiedades. Como valor de `label`, escriba **`Add To Cart`**.

Adición de un detector de eventos para activar la visualización de detalles de los regalos

A continuación añadirá un detector de eventos a la instancia `products_dg` de `DataGrid` para mostrar información sobre cada producto. Cuando el usuario haga clic en un producto de la sección `Gift Ideas`, aparecerá una ventana emergente con información sobre el producto.

- En el panel `Acciones` de la línea de tiempo principal, añada el código siguiente:

```
// Crear un detector para que el componente DataGrid detecte que la fila
del
// componente DataGrid ha cambiado
var dgListener:Object = new Object();
dgListener.change = function(evt:Object) {
    // Cuando cambie la fila actual en el componente DataGrid, iniciar una
    nueva
    // ventana emergente que muestre los detalles del producto.
    myWindow = mx.managers.PopUpManager.createPopUp(_root,
    mx.containers.Window, true, {title:evt.target.selectedItem.name,
    contentPath:"ProductForm", closeButton:true});
    // Establecer las dimensiones de la ventana emergente.
    myWindow.setSize(340, 210);
    // Definir un detector que cierre la ventana emergente cuando el
    usuario haga clic
    // en el botón de cierre.
    var closeListener:Object = new Object();
    closeListener.click = function(evt) {
        evt.target.deletePopUp();
    };
    myWindow.addEventListener("click", closeListener);
};
products_dg.addEventListener("change", dgListener);
```

Este código crea un nuevo detector de eventos denominado `dgListener` y crea instancias del componente `Window` que añadió anteriormente a la biblioteca. Se establece como título de la nueva ventana el nombre del producto. La ruta del contenido de la ventana se establece en el clip de película `ProductForm`. El tamaño de la ventana se establece en 340 x 210 píxeles.

El código también añade un botón de cierre para permitir que el usuario cierre la ventana después de ver la información.

Adición de código al clip de película ProductForm

A continuación añadirá código ActionScript al clip de película ProductForm que acaba de crear. El código ActionScript llena los componentes del clip de película con información sobre el regalo seleccionado e incluye un detector de eventos en el botón Add to Cart que añada el producto seleccionado al carrito.

Para más información sobre cómo trabajar con detectores de eventos, consulte “Utilización de detectores de eventos” en *Utilización de ActionScript en Flash*.

1. En la línea de tiempo del clip de película ProductForm, cree una nueva capa y asígnele el nombre **Actions**. Seleccione el primer fotograma en la capa Actions.
2. En el panel Acciones, añada el código siguiente:

```
// Crear un objeto que haga referencia al elemento de producto
// seleccionado en el componente DataGrid.
var thisProduct:Object = this._parent._parent.products_dg.selectedItem;
// Llenar las instancias description_ta de TextArea y price_lbl de Label
// con
// datos del producto seleccionado.
description_ta.text = thisProduct.description;
price_lbl.text = "<b>${thisProduct.price}"+
    "+thisProduct.priceQualifier"</b>";
// Cargar una imagen del producto desde el directorio de la aplicación.
image_ldr.load(thisProduct.image);
```

NOTA

El código incluye comentarios que describen su propósito. Es recomendable incluir comentarios como éstos en todo el código ActionScript que escriba. Así le resultará más fácil a usted (o a otro programador) entender el código más adelante, cuando retome el programa.

En primer lugar, el código define una variable para hacer referencia al producto seleccionado en el resto del programa. Si usa `thisProduct` no tendrá que hacer referencia al producto especificado con la ruta

```
this._parent._parent.products_dg.selectedItem.
```

A continuación, el código llena las instancias de `TextArea` y `Label` con los valores de las propiedades `description`, `price` y `priceQualifier` del objeto `thisProduct`. Estas propiedades corresponden a elementos del archivo `products.xml` que vinculó a la instancia `products_xml` con `XMLConnector` al principio del tutorial. En una fase posterior del tutorial vinculará entre sí instancias de los componentes `XMLConnector`, `DataSet` y `DataGrid`, y los elementos del archivo XML llenarán las otras dos instancias de componentes.

Por último, el código usa la propiedad `image` de la instancia del objeto `thisProduct` para cargar una imagen del producto en el componente `Loader`.

3. A continuación va a incluir un detector de eventos para añadir el producto al carrito cuando el usuario haga clic en el botón Add to Cart. (Añadirá código ActionScript a la línea de tiempo principal de la aplicación en una fase posterior del tutorial, a fin de crear una instancia de la clase Cart.) Añada el código siguiente:

```
var cartListener:Object = new Object();
cartListener.click = function(evt:Object) {
    var tempObj:Object = new Object();
    tempObj.quantity = evt.target._parent.quantity_ns.value;
    tempObj.id = thisProduct.id;
    tempObj.productObj = thisProduct;
    var theCart = evt.target._parent._parent.myCart;
    theCart.addProduct(tempObj.quantity, thisProduct);
};
addToCart_button.addEventListener("click", cartListener);
```

4. Haga clic en el botón Revisar sintaxis (la marca de verificación de color azul situada sobre el panel Script) para asegurarse de que no hay errores sintácticos en el código.

Puede revisar la sintaxis con frecuencia a medida que añade código a una aplicación. Los errores encontrados en el código se mostrarán en el panel Salida. Cuando se comprueba la sintaxis, sólo se comprueba el script actual; el resto de los scripts que puedan encontrarse en el archivo FLA no se comprueban. Para más información, consulte “Depuración de los scripts” en *Aprendizaje de ActionScript 2.0 en Flash*.

5. Haga clic en el botón de flecha situado en la esquina superior izquierda de la ventana de documento o seleccione Ver > Editar documento para salir del modo de edición de símbolos y volver a la línea de tiempo principal.

SUGERENCIA

Presione Control+S para guardar el trabajo y, a continuación Control+Intro (o seleccione Control > Probar película) para probar la aplicación. Si hace clic en una selección de regalo, debería aparecer una ventana con una imagen del regalo, acompañada de una descripción, el precio y un contador numérico que le permite seleccionar la cantidad de productos.

Creación de la página de finalización de la compra

Cuando el usuario hace clic en el botón Checkout de la pantalla principal, aparece la pantalla Checkout. La pantalla Checkout proporciona formularios en los que el usuario puede introducir los datos de facturación, de envío y de la tarjeta crédito. La pantalla de finalización de la compra tiene el siguiente aspecto:



The screenshot shows a checkout interface with a dark background. At the top left, the word "checkout" is written in a simple font. At the top right, the phrase "Fix Your Mistake" is written in a large, elegant, white cursive font. Below the title, there is a light-colored rectangular area containing three sections of a form, each with a blue header:

- 1. Billing Information**: This section contains five input fields: "First Name", "Last Name", "Country", "Province/State", and "City".
- 2. Shipping Information**: This section is currently hidden.
- 3. Credit Card Information**: This section is currently hidden.

At the bottom right of the form area, there is a small, rounded rectangular button labeled "Back".

La interfaz de finalización de la compra consta de componentes colocados en un fotograma clave, en el fotograma 10 de la aplicación. Usará el componente Accordion para crear la interfaz de finalización de la compra. El componente Accordion es un navegador que contiene una secuencia de elementos secundarios que se muestran de uno en uno. También añadirá una instancia del componente Button para crear un botón Back que permita a los usuarios volver a la pantalla principal.

En una fase posterior del tutorial, creará clips de película que se usarán como elementos secundarios en la instancia de Accordion y se mostrarán en los paneles Billing Information, Shipping Information y Credit Card Information.

1. En la línea de tiempo principal de la aplicación, mueva la cabeza lectora al fotograma 10 (denominado Checkout). Asegúrese de que la capa Form está seleccionada.
2. Inserte un fotograma clave vacío en el fotograma 10 de la capa Form (seleccione el fotograma y después seleccione Insertar > Línea de tiempo > Fotograma clave vacío).
3. Con el nuevo fotograma clave seleccionado, arrastre una instancia del componente Accordion desde el árbol User Interface del panel Componentes al escenario. En el inspector de propiedades, escriba **checkout_acc** como nombre de instancia. Establezca una anchura de **300** píxeles y una altura de **200** píxeles.
4. Arrastre una instancia del componente Button desde el árbol User Interface del panel Componentes a la esquina inferior derecha del escenario. En el inspector de propiedades, escriba **back_button** como nombre de la instancia. Haga clic en la ficha Parámetros y escriba **Back** como valor de la propiedad `label`.

Los paneles Billing Information, Shipping Information y Credit Card Information

Los paneles Billing Information, Shipping Information y Credit Card Information se crean con instancias de clip de película mostradas en la instancia del componente Accordion. Cada panel consta de dos clips de película anidados.

El clip de película principal contiene un componente ScrollPane, que se usa para mostrar contenido en un área desplazable. El clip de película secundario contiene componentes Label y TextInput en los que los usuarios pueden introducir sus datos personales, como el nombre, la dirección, etc. Utilizará el componente ScrollPane para mostrar el clip de película secundario de forma que el usuario pueda desplazarse por los campos de información.

Creación del panel Billing Information

Primero va a crear dos clips de película que mostrarán los campos del formulario Billing Information: un clip de película principal con la instancia del componente ScrollPane y un clip de película secundario con las instancias de los componentes Label y TextArea.

1. En el panel Biblioteca (Ventana > Biblioteca), haga clic en el menú de opciones del lado derecho de la barra de título y seleccione Nuevo símbolo.
2. En el cuadro de diálogo Crear un nuevo símbolo, escriba **checkout1_mc** como valor de Nombre y seleccione Clip de película como valor de Tipo.

3. Haga clic en el botón Avanzado. En Vinculación, seleccione Exportar para ActionScript, deje seleccionada la opción Exportar en primer fotograma y haga clic en Aceptar.
Se abrirá una ventana de documento para el nuevo símbolo en modo de edición de símbolos.
4. Arrastre una instancia del componente ScrollPane al escenario.
5. En el inspector de propiedades, escriba **checkout1_sp** como nombre de instancia. Establezca los valores de An. y Al. en **300, 135**. Establezca la coordenadas *x* e *y* en los valores **0, 0**.
6. Haga clic en la ficha Parámetros. Establezca el valor de la propiedad `contentPath` en **checkout1_sub_mc**.
Aparece el clip de película `checkout1_sub_mc` dentro del panel de desplazamiento y contiene los componentes Label y TextInput. A continuación creará este clip de película.
7. En el menú de opciones Biblioteca, seleccione Nuevo símbolo.
8. En el cuadro de diálogo Crear un nuevo símbolo, escriba **checkout1_sub_mc** como valor de Nombre y seleccione Clip de película como valor de Tipo.
9. Haga clic en el botón Avanzado. En Vinculación, seleccione Exportar para ActionScript, deje seleccionada la opción Exportar en primer fotograma y haga clic en Aceptar.
Se abrirá una ventana de documento para el nuevo símbolo en modo de edición de símbolos.
10. Arrastre seis instancias del componente Label al escenario. Como alternativa, puede arrastrar una instancia al escenario y hacer clic en Control (Windows) u Opción (Macintosh) para arrastarla al escenario para crear copias. Asigne a las instancias los siguientes nombres y posiciones:
 - Para la primera instancia, escriba **firstname_lbl** como nombre de la instancia y establezca las coordenadas *x* e *y* en **5, 5**. Haga clic en la ficha Parámetros y escriba **First Name** como valor de `text`.
 - Para la segunda instancia, escriba **lastname_lbl** como nombre de la instancia y establezca las coordenadas *x* e *y* en **5, 35**. Haga clic en la ficha Parámetros y escriba **Last Name** como valor de `text`.
 - Para la tercera instancia, escriba **country_lbl** como nombre de la instancia y establezca las coordenadas *x* e *y* en **5, 65**. Haga clic en la ficha Parámetros y escriba **Country** como valor de `text`.
 - Para la cuarta instancia, escriba **province_lbl** como nombre de la instancia y establezca las coordenadas *x* e *y* en **5, 95**. Haga clic en la ficha Parámetros y escriba **Province/ State** como valor de `text`.

- Para la quinta instancia, escriba `city_lbl` como nombre de la instancia y establezca las coordenadas x e y en `5, 125`. Haga clic en la ficha Parámetros y escriba `City` como valor de `text`.
 - Para la sexta instancia, escriba `postal_lbl` como nombre de la instancia y establezca las coordenadas x e y en `5, 155`. Haga clic en la ficha Parámetros y escriba `Postal/Zip Code` como valor de `text`.
11. Arrastre seis instancias del componente `TextInput` al escenario. Coloque una instancia de `TextInput` a la derecha de cada instancia de `Label`. Por ejemplo, las coordenadas x, y de la primera instancia de `TextInput` deben ser `105, 5`. Asigne a las instancias de `TextInput` los siguientes nombres:
- A la primera instancia, `billingFirstName_ti`.
 - A la segunda instancia, `billingLastName_ti`.
 - A la tercera instancia, `billingCountry_ti`.
 - A la cuarta instancia, `billingProvince_ti`.
 - A la quinta instancia, `billingCity_ti`.
 - A la sexta instancia, `billingPostal_ti`.
- A veces el contenido de un panel de desplazamiento puede verse recortado si está demasiado cerca del borde del panel. En los pasos siguientes añadirá un rectángulo de color blanco al clip de película `checkout1_sub_mc` para que las instancias de `Label` y `TextInput` se muestren de forma adecuada.
12. En la línea de tiempo, haga clic en el botón Insertar capa. Arrastre la nueva capa bajo la capa existente (la capa con el rectángulo debe estar en la parte inferior, de forma que el rectángulo no interfiera con la visualización del componente).
13. Seleccione el fotograma 1 de la nueva capa.
14. En el panel Herramientas, seleccione la herramienta Rectángulo. Establezca el Color de trazo en Ninguno y el Color de relleno en blanco.
- Haga clic en el control Color de trazo del panel Herramientas y después haga clic en el botón Ninguno (la muestra de color blanco atravesada por una línea roja). Haga clic en el control Color de relleno y después haga clic en la muestra de color blanco.
15. Arrastre para crear un rectángulo que se extienda más allá de los bordes inferior y derecho de las instancias de `Label` y `TextInput`.

Creación del panel Shipping Information

Los clips de película para el panel Shipping Information son similares a los del panel Billing Information. En este panel también añadirá un componente CheckBox, en el que los usuarios podrán llenar los campos del formulario Shipping Information con los mismos datos que especificaron en el panel Billing Information.

1. Siga las instrucciones anteriores (véase [“Creación del panel Billing Information” en la página 43](#)) para crear los clips de película para el panel Credit Card Information. Tenga en cuenta estas diferencias de nomenclatura:
 - Para el primer clip de película, especifique **checkout2_mc** como nombre del símbolo y **checkout2_sp** como nombre de la instancia. En la ficha Parámetros del inspector de propiedades, establezca el valor de la propiedad `contentPath` en **checkout2_sub_mc**.
 - Para el segundo clip de película, escriba **checkout2_sub_mc** como nombre del símbolo.
 - Para las instancias de TextInput, cambie “billing” por “shipping” en los nombres de instancia.
2. Con el clip de película `checkout2_sub_mc` abierto en modo de edición de símbolos, arrastre una instancia del componente CheckBox al escenario y colóquela justo encima de la primera instancia de Label.

Asegúrese de colocar esta instancia en la capa 1, junto con las demás instancias de componentes.
3. En el inspector de propiedades, escriba **sameAsBilling_ch** como nombre de la instancia.
4. Haga clic en la ficha Parámetros. Establezca el valor de la propiedad `label` en **Same As Billing Info**.

Creación del panel Credit Card Information

Los clips de película para el panel Credit Card Information también son similares a los de los paneles Billing Information y Shipping Information. No obstante, el clip de película anidado para el panel Credit Card Information tiene campos ligeramente distintos a los de los otros dos paneles, para el número de la tarjeta de crédito y los otros datos de la tarjeta.

1. Siga los pasos 1-9 de las instrucciones de Billing Information (véase [“Creación del panel Billing Information” en la página 43](#)) para crear los clips de película para el panel Credit Card Information. Tenga en cuenta estas diferencias de nomenclatura:
 - Para el primer clip de película, especifique **checkout3_mc** como nombre del símbolo y **checkout3_sp** como nombre de la instancia. En la ficha Parámetros del inspector de propiedades, establezca el valor de la propiedad `contentPath` en **checkout3_sub_mc**.
 - Para el segundo clip de película, escriba **checkout3_sub_mc** como nombre del símbolo.
2. Arrastre cuatro instancias del componente Label al escenario. Asigne a las instancias los siguientes nombres y posiciones:
 - Para la primera instancia, escriba **ccName_lbl** como nombre de la instancia y establezca las coordenadas x e y en 5, 5. Haga clic en la ficha Parámetros y escriba **Name On Card** como valor de `text`.
 - Para la segunda instancia, escriba **ccType_lbl** como nombre de la instancia y establezca las coordenadas x e y en 5, 35. Haga clic en la ficha Parámetros y escriba **Card Type** como valor de `text`.
 - Para la tercera instancia, escriba **ccNumber_lbl** como nombre de la instancia y establezca las coordenadas x e y en 5, 65. Haga clic en la ficha Parámetros y escriba **Card Number** como valor de `text`.
 - Para la cuarta instancia, escriba **ccExp_lbl** como nombre de la instancia y establezca las coordenadas x e y en 5, 95. Haga clic en la ficha Parámetros y escriba **Expiration** como valor de `text`.
3. Arrastre una instancia del componente TextInput al escenario y colóquela a la derecha de la instancia **ccName_lbl**. Asigne a la nueva instancia el nombre **ccName_ti**. Establezca la coordenadas x e y en los valores 105, 5. Establezca una anchura de 140.
4. Arrastre una instancia del componente ComboBox al escenario y colóquela a la derecha de la instancia **ccType_lbl**. Asigne a la nueva instancia el nombre **ccType_cb**. Establezca la coordenadas x e y en los valores 105, 35. Establezca una anchura de 140.
5. Arrastre otra instancia del componente TextInput al escenario y colóquela a la derecha de la instancia **ccNumber_lbl**. Asigne a la nueva instancia el nombre **ccNumber_ti**. Establezca la coordenadas x e y en los valores 105, 65. Establezca una anchura de 140.

6. Arrastre dos instancias del componente ComboBox al escenario. Coloque una a la derecha de la instancia `ccExp_lbl` y la otra a la derecha de la primera. Asigne a la primera nueva instancia el nombre `ccMonth_cb`. Establezca una anchura de **60** y las coordenadas *x* e *y* en **105, 95**. Asigne a la segunda instancia el nombre `ccYear_cb`. Establezca una anchura de **70** y las coordenadas *x* e *y* en **175, 95**.
7. Arrastre una instancia del componente Button al escenario y colóquela en la parte inferior del formulario, bajo la instancia `ccMonth_cb`. Asigne a la nueva instancia el nombre `checkout_button`. Establezca la coordenadas *x* e *y* en los valores **125, 135**. En la ficha Parámetros del inspector de propiedades, establezca el valor de la propiedad `label` en **Checkout**.
8. Siga las indicaciones de los pasos 14-15 de las instrucciones de Billing Information (véase “Creación del panel Billing Information” en la página 43) para añadir un rectángulo a la parte inferior del formulario.

Adición de un detector de eventos al botón Checkout

Ahora va a añadir código para mostrar la pantalla Checkout cuando el usuario haga clic en el botón Checkout.

- En el panel Acciones de la página principal, añada el código siguiente:

```
// Cuando se hace clic en el botón Checkout, ir a la etiqueta de
  fotograma "checkout".
var checkoutBtnListener:Object = new Object();
checkoutBtnListener.click = function(evt:Object) {
    evt.target._parent.gotoAndStop("checkout");
};
checkout_button.addEventListener("click", checkoutBtnListener);
```

Este código especifica que cuando el usuario haga clic en el botón Checkout, la cabeza lectora se moverá a la etiqueta Checkout de la línea de tiempo.

Adición de código para la pantalla Checkout

Ya está preparado para añadir código a la pantalla Checkout de la aplicación, en el fotograma 10 de la línea de tiempo principal. Este código procesa los datos introducidos por el usuario en los paneles Billing Information, Shipping Information y Credit Card Information que creó anteriormente con el componente Accordion y otros componentes.

1. En la línea de tiempo, seleccione el fotograma 10 en la capa Acciones e inserte un fotograma clave vacío (seleccione Insertar > Línea de tiempo > Fotograma clave vacío)
2. Abra el panel Acciones (F9).
3. En el panel Acciones, añada el código siguiente:

```
stop();
import mx.containers.*;

// Definir el componente Accordion en el escenario.
var checkout_acc:Accordion;
```

4. A continuación debe añadir el primer elemento secundario a la instancia del componente Accordion para aceptar los datos de facturación del usuario. Añada el código siguiente:

```
// Definir los elementos secundarios del componente Accordion.
var child1 = checkout_acc.createChild("checkout1_mc", "child1_mc",
    {label:"1. Billing Information"});
var thisChild1 = child1.checkout1_sp.spContentHolder;
```

La primera línea llama al método `createChild()` del componente Accordion y crea una instancia del símbolo de clip de película `checkout1_mc` (que creó anteriormente) con el nombre de instancia `child1_mc` y la etiqueta “1. Billing Information”. La segunda línea de código crea un método abreviado a una instancia del componente ScrollPane incorporada.

5. Cree el segundo elemento secundario para la instancia de Accordion, para aceptar los datos de envío:

```
/* Añadir el segundo elemento secundario al componente Accordion.
Añada un detector de eventos para la instancia sameAsBilling_ch de
CheckBox.
Esto copia los valores del formulario desde el primer elemento secundario
al segundo elemento secundario. */
var child2 = checkout_acc.createChild("checkout2_mc", "child2_mc",
    {label:"2. Shipping Information"});
var thisChild2 = child2.checkout2_sp.spContentHolder;
var checkBoxListener:Object = new Object();
checkBoxListener.click = function(evt:Object) {
    if (evt.target.selected) {
        thisChild2.shippingFirstName_ti.text =
            thisChild1.billingFirstName_ti.text;
        thisChild2.shippingLastName_ti.text =
            thisChild1.billingLastName_ti.text;
```

```

        thisChild2.shippingCountry_ti.text =
        thisChild1.billingCountry_ti.text;
        thisChild2.shippingProvince_ti.text =
        thisChild1.billingProvince_ti.text;
        thisChild2.shippingCity_ti.text = thisChild1.billingCity_ti.text;
        thisChild2.shippingPostal_ti.text =
        thisChild1.billingPostal_ti.text;
    }
};
thisChild2.sameAsBilling_ch.addEventListener("click", checkboxListener);

```

Las primeras dos líneas de código son similares al código para crear el elemento secundario de Billing Information: se crea una instancia del símbolo del clip de película checkout2_mc con el nombre de instancia child2_mc y la etiqueta “2. Shipping Information”. La segunda línea de código crea un método abreviado a una instancia del componente ScrollPane incorporada.

Empezando por la tercera línea de código, puede añadir un detector de eventos a la instancia de CheckBox. Si el usuario hace clic en la casilla de verificación, la información de envío usa los datos que el usuario especificó en el panel Billing Information.

6. A continuación, cree un tercer elemento secundario para la instancia de Accordion (para la información de la tarjeta crédito):

```

// Definir el tercer elemento secundario de Accordion.
var child3 = checkout_acc.createChild("checkout3_mc", "child3_mc",
    {label:"3. Credit Card Information"});
var thisChild3 = child3.checkout3_sp.spContentHolder;

```

7. Añada este código para crear instancias de ComboBox para el mes, el año y el tipo de la tarjeta de crédito, y llénelas con una matriz definida estáticamente:

```

/* Establecer los valores de las tres instancias de ComboBox en el
    escenario:
ccMonth_cb, ccYear_cb and ccType_cb */
thisChild3.ccMonth_cb.labels = ["01", "02", "03", "04", "05", "06",
    "07", "08", "09", "10", "11", "12"];
thisChild3.ccYear_cb.labels = [2004, 2005, 2006, 2007, 2008, 2009,
    2010];
thisChild3.ccType_cb.labels = ["VISA", "MasterCard", "American Express",
    "Diners Club"];

```

8. Por último, añada el código siguiente para añadir detectores de eventos al botón Checkout y el botón Back. Cuando el usuario haga clic en el botón Checkout, el objeto detector copiará los campos del formulario de los paneles Billing Information, Shipping Information y Credit Card Information a un objeto LoadVars que se envía al servidor. (La clase LoadVars permite enviar todas las variables de un objeto a una URL especificada.) Cuando el usuario haga clic en el botón Back, la aplicación volverá a la pantalla principal.

```

/* Crear un detector para la instancia checkout_button de Button.

```

```

Este detector envía todas las variables de formulario al servidor cuando
    el usuario hace clic en el botón Checkout. */
var checkoutListener:Object = new Object();
checkoutListener.click = function(evt:Object){
    evt.target.enabled = false;
    /* Crear dos instancias del objeto LoadVars, que envíe variables al
    servidor remoto
    y reciba resultados del mismo. */
    var response_lv:LoadVars = new LoadVars();
    var checkout_lv:LoadVars = new LoadVars();
    checkout_lv.billingFirstName = thisChild1.billingFirstName_ti.text;
    checkout_lv.billingLastName = thisChild1.billingLastName_ti.text;
    checkout_lv.billingCountry = thisChild1.billingCountry_ti.text;
    checkout_lv.billingProvince = thisChild1.billingProvince_ti.text;
    checkout_lv.billingCity = thisChild1.billingCity_ti.text;
    checkout_lv.billingPostal = thisChild1.billingPostal_ti.text;
    checkout_lv.shippingFirstName = thisChild2.shippingFirstName_ti.text;
    checkout_lv.shippingLastName = thisChild2.shippingLastName_ti.text;
    checkout_lv.shippingCountry = thisChild2.shippingCountry_ti.text;
    checkout_lv.shippingProvince = thisChild2.shippingProvince_ti.text;
    checkout_lv.shippingCity = thisChild2.shippingCity_ti.text;
    checkout_lv.shippingPostal = thisChild2.shippingPostal_ti.text;
    checkout_lv.ccName = thisChild3.ccName_ti.text;
    checkout_lv.ccType = thisChild3.ccType_cb.selectedItem;
    checkout_lv.ccNumber = thisChild3.ccNumber_ti.text;
    checkout_lv.ccMonth = thisChild3.ccMonth_cb.selectedItem;
    checkout_lv.ccYear = thisChild3.ccYear_cb.selectedItem;

    /* Enviar las variables de la instancia checkout_lv de LoadVars al
    script remoto en el servidor.
    Guardar los resultados en la instancia response_lv. */
    checkout_lv.sendAndLoad("http://www.flash-mx.com/mm/firstapp/
    cart.cfm", response_lv, "POST");
    response_lv.onLoad = function(success:Boolean) {
        evt.target.enabled = true;
    };
};
thisChild3.checkout_button.addEventListener("click", checkoutListener);
cart_mc._visible = false;
var backListener:Object = new Object();
backListener.click = function(evt:Object) {
    evt.target._parent.gotoAndStop("home");
}
back_button.addEventListener("click", backListener);

```

Prueba de la aplicación

¡Enhorabuena! Ha finalizado la aplicación. Presione Control+S para guardar el trabajo y, a continuación Control+Intro (o seleccione Control >Probar película) para probar la aplicación.

Visualización de la aplicación finalizada

En el caso de que no haya podido completar correctamente el tutorial, puede ver una versión que funcione de la aplicación finalizada. Puede encontrar este archivo de Flash (FLA) de inicio, `first_app_start fla`, y el archivo finalizado, `first_app fla`, en la carpeta Samples del disco duro:

- En Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\ComponentsApplication.
- En Macintosh: *Disco duro de Macintosh*\Applications\Macromedia Flash 8\Samples and Tutorials\Samples\Components\ComponentsApplication.

Para ver el archivo FLA de la aplicación, abra el archivo `first_app fla` en la carpeta `components_application`.

Puede comparar estos archivos con los suyos para detectar posibles errores.

Todos los componentes usados en la aplicación aparecen en la biblioteca (junto con archivos de gráficos y otros activos usados para crear la aplicación). Algunos componentes aparecen como instancias en el escenario. Se hace referencia a otros componentes en el código ActionScript, pero no aparecen hasta el tiempo de ejecución.

En este capítulo se usarán varios archivos de Macromedia Flash (FLA) y archivos de clase de ActionScript para aprender a añadir componentes a un documento y establecer sus propiedades. También se explican algunos temas avanzados como el uso de las sugerencias para el código, la forma de crear un desplazamiento personalizado de la selección, la administración de la profundidad de componentes y la actualización de componentes de la versión 1 de la arquitectura de componentes de Macromedia a la versión 2.

Los archivos usados en este capítulo son TipCalculator.fla y TipCalculator.swf. Se instalan en las siguientes ubicaciones del disco duro:

- (Windows) Archivos de programa\Macromedia\FIash8\Samples and Tutorials\Samples\Components\TipCalculator
- (Macintosh) Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/TipCalculator

Este capítulo contiene los siguientes temas:

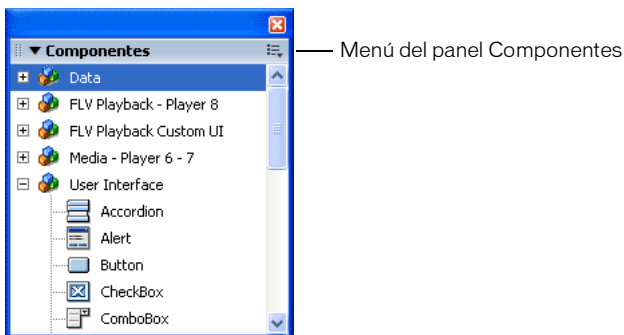
Panel Componentes	54
Adición de componentes a documentos de Flash	54
Componentes del panel Biblioteca	58
Definición de parámetros de componentes	59
Ajuste del tamaño de los componentes	61
Eliminación de componentes de documentos de Flash	62
Utilización de las sugerencias para el código	62
Creación de un desplazamiento personalizado de la selección	63
Administración de la profundidad del componente en un documento	64
Componentes en previsualización dinámica	65
Utilización de un precargador con componentes	65
Carga de componentes	67
Actualización de componentes de la versión 1 de la arquitectura de componentes a la versión 2	68

Panel Componentes

Todos los componentes de la configuración de nivel de usuario y del directorio Components se muestran en el panel Componentes. (Para más información sobre este directorio, consulte [“Ubicación de almacenamiento de los archivos de componentes” en la página 14.](#))

Para mostrar el panel Componentes:

- Seleccione Ventana > Componentes.



Para mostrar componentes que se instalaron después de iniciar Flash:

1. Seleccione Ventana > Componentes.
2. Seleccione Volver a cargar en el menú emergente del panel Componentes.

Adición de componentes a documentos de Flash

Si arrastra un componente desde el panel Componentes al escenario, se añadirá un símbolo de clip compilado (SWC) al panel Biblioteca. Tras arrastrar un símbolo SWC a la biblioteca, puede arrastrar varias instancias al escenario. También puede añadir ese componente a un documento en tiempo de ejecución mediante el método `UIObject.createClassObject()` de `ActionScript`.

NOTA

Los componentes **Menu** y **Alert** son dos excepciones y no se puede crear instancias de los mismos mediante `UIObject.createClassObject()`. En su lugar, se utiliza el método `show()`.

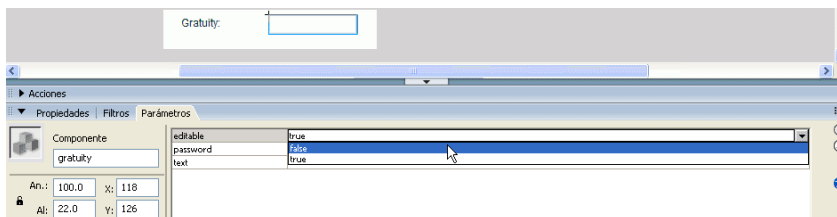
Adición de componentes durante la edición

Puede añadir un componente a un documento a través del panel Componentes y añadir después nuevas instancias del componente al documento arrastrando el componente desde el panel Biblioteca al escenario. Puede definir propiedades para instancias adicionales en la ficha Parámetros del inspector de propiedades o en la misma ficha del inspector de componentes.

Para añadir un componente a un documento de Flash a través del panel Componentes:

1. Seleccione Ventana > Componentes.
2. Siga uno de estos procedimientos:
 - Arrastre un componente desde el panel Componentes al escenario.
 - Haga doble clic en un componente del panel Componentes.
3. Si el componente es un archivo FLA (todos los componentes instalados de la versión 2 son archivos SWC) y si ha editado aspectos para otra instancia del mismo componente o para un componente que comparte aspectos con el componente que va a añadir, siga uno de estos procedimientos:
 - Seleccione No reemplazar elementos ya existentes para mantener los aspectos editados y aplicarlos al nuevo componente.
 - Si desea reemplazar todos los aspectos por aspectos predeterminados, seleccione Reemplazar elementos ya existentes. El nuevo componente y todas sus versiones anteriores, o las de los componentes que comparten sus aspectos, utilizarán los aspectos predeterminados.
4. Seleccione el componente en el escenario.
5. Seleccione Ventana > Propiedades > Propiedades.
6. En el inspector de propiedades, introduzca un nombre para la instancia de componente.
7. Haga clic en la ficha Parámetros y especifique parámetros para la instancia.

En la siguiente ilustración se muestra el inspector de propiedades para el componente TextInput que está en el archivo TipCalculator.fla de ejemplo (instalado en Flash 8/Samples and Tutorials/Samples/Components/TipCalculator).



Para más información, consulte [“Definición de parámetros de componentes” en la página 59](#).

8. Cambie el tamaño del componente como desee, editando los valores de anchura y altura. Para más información sobre cómo cambiar el tamaño de tipos de componente específicos, consulte las entradas referentes al componente en cuestión en *Referencia del lenguaje de componentes*.
9. Si desea cambiar el color y el formato del texto de un componente, siga uno o más de estos procedimientos:
 - Defina o cambie un valor de propiedad de estilo específico para una instancia de componente mediante el método `setStyle()` disponible para todos los componentes. Para más información, consulte [UIObject.setStyle\(\) en la página 1450](#).
 - Edite varias propiedades en la declaración de estilo global asignada a todos los componentes de la versión 2.
 - Cree una declaración de estilo personalizada para instancias de componente específicas. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 86](#).
10. Si desea personalizar la apariencia del componente, siga uno de estos procedimientos:
 - Aplique un tema (véase [“Temas” en la página 115](#)).
 - Edite los aspectos de un componente (véase [“Aplicación de aspectos a los componentes” en la página 102](#)).

Adición de componentes en tiempo de ejecución mediante ActionScript

En las instrucciones de esta sección se presupone que el usuario posee un conocimiento intermedio o avanzado de ActionScript.

Use el método `createClassObject()` (que la mayoría de los componentes heredan de la clase `UIObject`) para añadir componentes dinámicamente a una aplicación Flash. Por ejemplo, puede añadir componentes que crean un diseño de página basado en las preferencias establecidas por el usuario (como en la página de inicio de un portal Web).

Los componentes de la versión 2 que se instalan con Flash residen en directorios de paquete. Para más información, consulte “Paquetes” en *Aprendizaje de ActionScript 2.0 en Flash*. Si añade un componente al escenario durante la edición, puede hacer referencia al componente usando simplemente su nombre de instancia (por ejemplo, `myButton`). No obstante, si añade un componente a una aplicación mediante código ActionScript (en tiempo de ejecución), debe especificar su nombre de clase completo (por ejemplo, `mx.controls.Button`) o importar el paquete mediante la sentencia `import`.

Por ejemplo, para escribir código ActionScript que haga referencia a un componente `Alert`, puede utilizar la sentencia `import` para hacer referencia a la clase, de la manera siguiente:

```
import mx.controls.Alert;
Alert.show("The connection has failed", "Error");
```

Como alternativa, puede utilizar la ruta completa del paquete, de la manera siguiente:

```
mx.controls.Alert.show("The connection has failed", "Error");
```

Para más información, consulte “Importación de archivos de clases” en *Aprendizaje de ActionScript 2.0 en Flash*.

Puede utilizar métodos ActionScript para definir otros parámetros para componentes añadidos dinámicamente. Para más información, consulte *Referencia del lenguaje de componentes*.

NOTA

Para añadir un componente a un documento en tiempo de ejecución, éste debe estar en la biblioteca cuando se compile el archivo SWF. Para añadir un componente a la biblioteca, arrastre el icono del componente desde el panel Componentes a la biblioteca. Además, si carga un clip de película que contiene un componente del que se ha creado dinámicamente una instancia (mediante código ActionScript) en otro clip de película, el clip de película principal debe tener el componente en la biblioteca cuando se compile el archivo SWF.

Para añadir un componente a un documento de Flash mediante ActionScript:

1. Arrastre un componente del panel Componentes a la biblioteca del documento actual.

NOTA

Los componentes están definidos de forma predeterminada en Exportar en primer fotograma (haga clic con el botón derecho del ratón en Windows o con la tecla Control presionada en Macintosh y seleccione la opción de menú Vinculación para ver la configuración de Exportar en primer fotograma). Si desea utilizar un precargador para una aplicación que contenga componentes, debe cambiar el fotograma de exportación. Consulte [“Utilización de un precargador con componentes” en la página 65](#) para obtener instrucciones.

2. Seleccione en la línea de tiempo el fotograma donde desea añadir el componente.
3. Abra el panel Acciones si aún no está abierto.

4. Llame a `createClassObject()` para crear la instancia de componente en tiempo de ejecución.

Este método puede llamarse por sí solo, o bien a partir de una instancia de componente. El método `createClassObject()` usa los siguientes parámetros: un nombre de clase de componente, un nombre de instancia para la nueva instancia, una profundidad y un objeto de inicialización opcional que puede utilizar para establecer propiedades en tiempo de ejecución.

Puede especificar el paquete de la clase en el parámetro de nombre de la clase, como en el siguiente ejemplo:

```
createClassObject(mx.controls.CheckBox, "cb", 5, {label:"Check Me"});
```

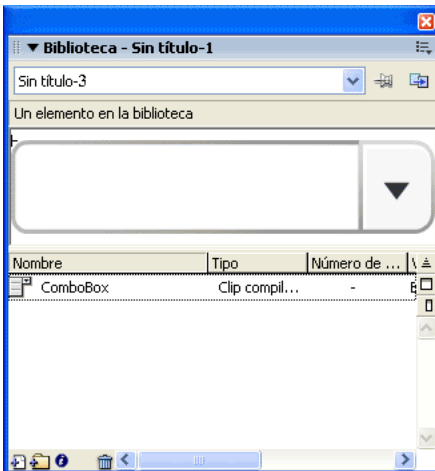
Como alternativa, puede importar el paquete de clase, como en el siguiente ejemplo:

```
import mx.controls.CheckBox;  
createClassObject(CheckBox, "cb", 5, {label:"Check Me"});
```

Para más información, consulte [UIObject.createClassObject\(\) en la página 1428](#) y [Capítulo 4, “Gestión de eventos de componentes”, en la página 69](#).

Componentes del panel Biblioteca

Cuando se añade un componente a un documento, el componente aparece como un símbolo de clip compilado (archivo SWC) en el panel Biblioteca.



Componente ComboBox del panel Biblioteca

Puede añadir más instancias de un componente arrastrando el icono del componente desde la biblioteca al escenario.

Para más información sobre clips compilados, consulte [“Clips compilados y archivos SWC” en la página 21](#).

Definición de parámetros de componentes

Todos los componentes tienen parámetros que se pueden definir para cambiar su aspecto y comportamiento. Un parámetro es una propiedad que aparece en el inspector de propiedades y en el inspector de componentes. Las propiedades que se utilizan con más frecuencia aparecen como parámetros de edición; las otras deben establecerse mediante código ActionScript. Todos los parámetros que se pueden definir durante la edición también se pueden definir mediante código ActionScript. Los valores de los parámetros establecidos mediante ActionScript sustituyen cualquier otro valor que se haya establecido durante la edición.

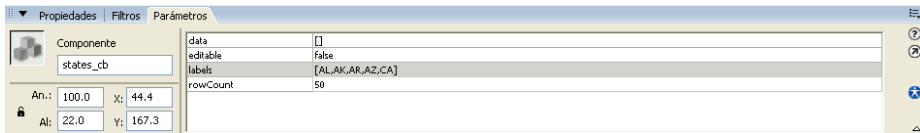
Todos los componentes de interfaz de usuario de la versión 2 heredan propiedades y métodos de las clases `UIObject` y `UIComponent`; éstas son las propiedades y los métodos que usan todos los componentes, como `UIObject.setSize()`, `UIObject.setStyle()`, `UIObject.x` y `UIObject.y`. Asimismo, cada componente tiene propiedades y métodos exclusivos, algunos de los cuales están disponibles como parámetros de edición. Por ejemplo, el componente `ProgressBar` tiene una propiedad `percentComplete` (`ProgressBar.percentComplete`) y el componente `NumericStepper` dispone de las propiedades `nextValue` y `previousValue` (`NumericStepper.nextValue`, `NumericStepper.previousValue`).

Puede configurar los parámetros de una instancia de componente mediante el inspector de componentes o el inspector de propiedades (no importa qué panel use).

Para introducir un nombre de instancia para un componente en el inspector de propiedades:

1. Seleccione Ventana > Propiedades > Propiedades.
2. Seleccione una instancia de componente en el escenario.
3. Introduzca un nombre de instancia en el cuadro de texto bajo la palabra *Componente*.

Es recomendable añadir al nombre de la instancia un sufijo que indique el tipo de componente que es; esto facilitará la lectura del código ActionScript. En este ejemplo, el nombre de la instancia es **states_cb** (el componente es un cuadro combinado que muestra la lista de estados de EE.UU.).



Para introducir parámetros para una instancia de componente en el inspector de componentes:

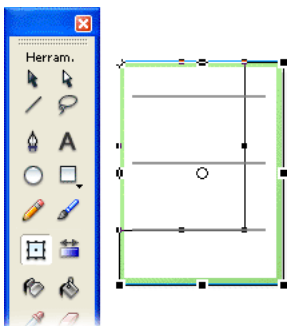
1. Seleccione Ventana > Inspector de componentes.
2. Seleccione una instancia de componente en el escenario.
3. Para introducir los parámetros, haga clic en la ficha Parámetros.



4. Para introducir o ver vinculaciones o esquemas para un componente, haga clic en sus fichas respectivas. Para más información, consulte “Integración de datos (sólo para Flash Professional)” en *Utilización de Flash*.

Ajuste del tamaño de los componentes

Utilice la herramienta Transformación libre o el método `setSize()` para cambiar el tamaño de las instancias de componente.



Ajuste del tamaño del componente Menu en el escenario mediante la herramienta Transformación libre

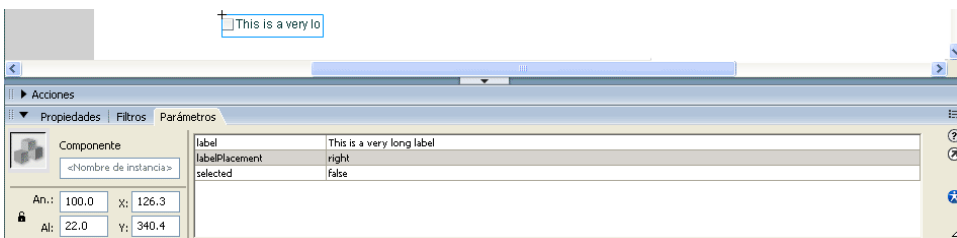
Puede llamar al método `setSize()` desde cualquier instancia de componente (véase [UIObject.setSize\(\) en la página 1447](#)) para cambiar el tamaño de dicha instancia. El código siguiente cambia el tamaño del componente `TextArea` a 200 píxeles de ancho por 300 píxeles de alto:

```
myTextArea.setSize(200, 300);
```

NOTA

Si utiliza las propiedades `_width` y `_height` de `ActionScript` para ajustar la anchura y la altura de un componente, cambiará el tamaño del componente, pero no el diseño del contenido. Esto puede provocar la distorsión del componente al reproducir la película.

Un componente no cambia de tamaño automáticamente para ajustarse a su etiqueta. Si una instancia de componente que se ha añadido a un documento no es lo suficientemente grande para mostrar la etiqueta, el texto de la etiqueta aparece recortado. Debe cambiar el tamaño del componente para que quepa en su etiqueta.



Una etiqueta recortada para el componente CheckBox

Para más información sobre el ajuste del tamaño de los componentes, consulte sus respectivas entradas en *Referencia del lenguaje de componentes*.

Eliminación de componentes de documentos de Flash

Para eliminar instancias de un componente de un documento de Flash, debe borrar el componente de la biblioteca eliminando el icono del clip compilado. No basta con eliminar el componente del escenario.

Para eliminar un componente de un documento:

1. En el panel Biblioteca, seleccione el símbolo de clip compilado (SWC).
2. Haga clic en el botón Eliminar situado en la parte inferior del panel Biblioteca, o seleccione Eliminar en el menú Opciones de Biblioteca.
3. En el cuadro de diálogo Eliminar, haga clic en Eliminar para confirmar la operación.

Utilización de las sugerencias para el código

Cuando utiliza ActionScript 2.0, puede usar tipos de datos estrictos para variables basadas en clases incorporadas, incluidas las clases de componente. De este modo, el editor de ActionScript muestra las sugerencias para el código para la variable. Por ejemplo, suponga que escribe lo siguiente:

```
import mx.controls.CheckBox;  
var myCheckBox:CheckBox;  
myCheckBox.
```

En cuanto escriba el punto a continuación de `myCheckBox`, Flash mostrará una lista de métodos y propiedades disponibles para componentes `CheckBox`, porque la variable está declarada como de tipo `CheckBox`. Para más información, consulte “Asignación de tipos de datos y “strict data typing”” en “Utilización de las sugerencias para el código” en *Aprendizaje de ActionScript 2.0 en Flash*.

Creación de un desplazamiento personalizado de la selección

Cuando un usuario presiona la tecla Tabulador para navegar por una aplicación de Flash o hace clic en una aplicación, la clase FocusManager determina qué componente se selecciona (para más información, consulte [FocusManager, clase](#) en *Referencia del lenguaje de componentes*). No es preciso añadir una instancia de FocusManager a una aplicación, ni escribir código para activar Focus Manager.

Si se selecciona un objeto RadioButton, Focus Manager examina el objeto, así como todos los objetos con el mismo valor de groupName, y define la selección del objeto estableciendo el valor true para la propiedad selected.

Cada componente Window modal contiene una instancia de Focus Manager, de forma que los controles de esa ventana se conviertan en su propio grupo de tabulación. Esto evita que un usuario desplace accidentalmente la selección a componentes de otras ventanas al presionar la tecla Tabulador.

Para que la selección pueda desplazarse por la aplicación, establezca la propiedad tabIndex de los componentes que deban seleccionarse (botones incluidos). Cuando un usuario presiona la tecla Tabulador, la clase FocusManager busca un objeto activado cuyo valor de tabIndex sea mayor que el valor actual de tabIndex. Cuando la clase FocusManager alcanza el valor más alto de la propiedad tabIndex, vuelve a 0. Por ejemplo, en el código siguiente, el objeto comment (probablemente un componente TextArea) recibe primero la selección; a continuación, la instancia okButton recibe la selección:

```
var comment:mx.controls.TextArea;
var okButton:mx.controls.Button;
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 usa los niveles de profundidad (orden z). Los niveles de profundidad se configuran 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 el orden z final.

Para seleccionar un componente de una aplicación, llame a focusManager.setFocus().

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 en el código siguiente:

```
focusManager.defaultPushButton = okButton;
```

La [Clase FocusManager \(API\)](#) sustituye el rectángulo de selección de Flash Player predeterminado y dibuja un rectángulo de selección personalizado con esquinas redondeadas. Para más información sobre cómo crear un esquema de selección en una aplicación Flash, consulte [FocusManager, clase](#) en *Referencia del lenguaje de componentes*.

Administración de la profundidad del componente en un documento

Si desea colocar un componente delante (o detrás) de otro objeto de una aplicación, debe utilizar la [DepthManager, clase](#) descrita en *Referencia del lenguaje de componentes*. Los métodos de la clase DepthManager permiten colocar componentes de interfaz de usuario en un orden *relativo* apropiado (por ejemplo, un cuadro combinado se despliega delante de otros componentes, aparecen puntos de inserción delante de todo o hay cuadros de diálogo flotando sobre el contenido, etc.).

Depth Manager sirve para dos cosas principalmente: administrar las asignaciones de profundidad relativa en cualquier documento y administrar las profundidades reservadas en la línea de tiempo raíz para servicios del sistema como, por ejemplo, el cursor y las sugerencias.

Para utilizar Depth Manager, llame a sus métodos.

El código siguiente coloca la instancia del componente `loader` bajo el componente `button` (y en el archivo SWF publicado aparecerá “debajo” del botón, si se superponen):

```
loader.setDepthBelow(button);
```

NOTA

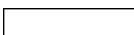
También es posible gestionar las profundidades relativas mediante las capas y las opciones de menú Modificar > Organizar en el documento. Los componentes siguen las mismas reglas para la gestión de la profundidad en tiempo de ejecución y utilizan las capas y las opciones de organización de igual forma que los clips de película.

Componentes en previsualización dinámica

La función Previsualización dinámica, que se activa de forma predeterminada, permite ver los componentes del escenario tal como aparecerán en el contenido de Flash publicado, con su tamaño aproximado. La previsualización dinámica refleja parámetros diferentes para componentes diferentes. Para obtener información sobre qué parámetros de componente se reflejan en la previsualización dinámica, consulte la entrada de cada componente en *Referencia del lenguaje de componentes*.



Componente Button con la previsualización dinámica activada



Componente Button con la previsualización dinámica desactivada

En la previsualización dinámica, los componentes no son funcionales. Para probar la funcionalidad de los componentes, utilice el comando Control > Probar película.

Para activar o desactivar la función Previsualización dinámica:

- Seleccione Control > Activar vista previa dinámica. Si aparece una marca de verificación junto a la opción, indica que está activada.

Utilización de un precargador con componentes

La precarga implica cargar algunos de los datos de un archivo SWF antes de que el usuario inicie la interacción con el archivo. Los componentes y las clases están definidos de forma predeterminada para exportar en el primer fotograma del documento que contiene los componentes. Como las clases y los componentes son los primeros datos que se cargan, puede tener problemas para implementar una barra de progreso o para cargar una animación. En concreto, es probable que los componentes y las clases se carguen antes de la barra de progreso pero desee que la barra refleje el progreso de la carga de todos los datos, incluidas las clases. Por lo tanto, debería cargar las clases después de otras partes del archivo SWF pero antes de utilizar los componentes.

Para ello, cuando cree un precargador personalizado para una aplicación que contenga componentes, establezca la configuración de publicación del archivo de forma que se exporten todas las clases en el fotograma que contiene los componentes. Para ver una lista de todos los componentes de los temas Halo y Sample que tienen sus elementos establecidos en Exportar en primer fotograma, consulte [“Cambio de la configuración de exportación” en la página 124](#).

Para cambiar el fotograma de exportación de todas las clases:

1. Seleccione Archivo > Configuración de publicación.
2. En la ficha Flash del cuadro de diálogo Configuración de publicación, asegúrese de que la versión de ActionScript establecida es ActionScript 2.0.
3. Haga clic en el botón Configuración situado a la derecha de la versión de ActionScript.
4. En Configuración de ActionScript 2.0, cambie el número especificado en el cuadro de texto Fotograma de exportación para clases por el fotograma en el que aparecen por primera vez los componentes.

No puede utilizar ninguna de las clases hasta que la cabeza lectora alcance el fotograma donde ha elegido que deben cargarse. Dado que los componentes necesitan clases para poder funcionar, debe cargar componentes después del fotograma especificado para cargar clases. Si exporta las clases en el fotograma 3, no podrá utilizar ninguna de las clases hasta que la cabeza lectora alcance el fotograma 3 y cargue los datos.

Si desea precargar un archivo que utiliza componentes, debe precargar también los componentes en el archivo SWF. Para ello, debe establecer la exportación de componentes en un fotograma distinto en el archivo SWF.

Para cambiar el fotograma en el que se exportan los componentes:

1. Seleccione Ventana > Biblioteca, para abrir el panel Biblioteca.
2. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el componente de la biblioteca.
3. Seleccione Vinculación en el menú contextual.
4. Desactive la opción Exportar en primer fotograma.
5. Haga clic en Aceptar.
6. Seleccione Archivo > Configuración de publicación.
7. Seleccione la ficha Flash y haga clic en el botón Configuración.
8. Introduzca un número en el cuadro de texto Fotograma de exportación para clases y haga clic en Aceptar. Las clases se cargarán en este fotograma.
9. Haga clic en Aceptar para cerrar el cuadro de diálogo Configuración de publicación.

Si los componentes no se cargan en el primer fotograma, puede crear una barra de progreso personalizada para el primer fotograma del archivo SWF. No haga referencia a ningún componente en el código ActionScript ni incluya componentes en el escenario hasta que haya cargado las clases para el fotograma especificado en el paso 7.

NOTA

Los componentes deben exportarse después de las clases de ActionScript que utilizan.

Carga de componentes

Si carga componentes de la versión 2 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*.

Actualización de componentes de la versión 1 de la arquitectura de componentes a la versión 2

Los componentes de la versión 2 se diseñaron para satisfacer varios estándares de Internet (relativos a eventos [www.w3.org/TR/DOM-Level-3-Events/events.html], estilos, directivas de captador/definidor, etc.) y son muy distintos de sus homólogos de la versión 1 que se lanzaron con Macromedia Flash MX y en los DRK que se lanzaron antes que Macromedia Flash MX 2004. Los componentes de la versión 2 tienen interfaces API distintas y se programaron en ActionScript 2.0. Por tanto, si se juntan componentes de la versión 1 y de la versión 2 en una aplicación, el comportamiento es impredecible. Para más información sobre cómo actualizar los componentes de la versión 1 para utilizar la gestión de eventos, los estilos y el acceso captador/definidor a las propiedades en lugar de métodos de la versión 2, consulte el [Capítulo 6, “Creación de componentes”, en la página 133](#).

Las aplicaciones Flash que contienen componentes de la versión 1 funcionan correctamente en Flash Player 6 y Flash Player 7, cuando se publican para Flash Player 6 o Flash Player 6 (6.0.65.0). Si desea actualizar sus aplicaciones para que funcionen cuando se publiquen para Flash Player 7, debe convertir el código para poder usar tipos de datos estrictos. Para más información, consulte “Escritura de archivos de clases personalizadas” en *Aprendizaje de ActionScript 2.0 en Flash*.

Gestión de eventos de componentes

Cada componente tiene eventos que se difunden cuando un usuario interactúa con él (por ejemplo, los eventos `click` y `change`) o cuando sucede algo significativo en el componente (por ejemplo, el evento `load`). Para gestionar un evento, escriba código ActionScript que se ejecute cuando se produzca el evento.

Cada componente difunde su propio conjunto de eventos. Este conjunto incluye los eventos de cualquier clase de la que herede el componente. Esto significa que todos los componentes, con la excepción de los componentes multimedia, heredan eventos de las clases `UIObject` y `UIComponent`, ya que son las clases base de la arquitectura de la versión 2. Si desea ver la lista de eventos que difunde un componente, vea la entrada del componente y las entradas de sus clases antecesoras en *Referencia del lenguaje de componentes*.

En este capítulo se usan varias versiones de una aplicación Macromedia Flash sencilla, TipCalculator, con la que aprenderá a gestionar eventos de componente. Los archivos FLA y SWF se instalan con Flash en:

- En Windows: carpeta `C:\Archivos de programa\Macromedia\Flex8\Samples and Tutorials\Samples\Components\TipCalculator`.
- En Macintosh: carpeta `Disco duro/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/TipCalculator`.

Este capítulo contiene las siguientes secciones:

Utilización de detectores para gestionar eventos	70
Delegación de eventos	78
El objeto de evento	82
Utilización del controlador de eventos <code>on()</code>	83

Utilización de detectores para gestionar eventos

La versión 2 de la arquitectura de componentes tiene un modelo de eventos de difusor/detector. (un *difusor* también se denomina a veces *distribuidor*). Es importante comprender los siguientes puntos clave del modelo:

- Todo los eventos se difunden a través de una instancia de una clase de componente. (La instancia del componente es el *difusor*.)
- Un *detector* puede ser una función o un objeto. Si el detector es un objeto, debe tener una función callback definida. El detector *gestiona* el evento; esto significa que la función callback se ejecuta cuando se produce el evento.
- Para registrar un detector en un difusor, llame al método `addEventListener()` desde el difusor. Utilice la siguiente sintaxis:

```
componentInstance.addEventListener("eventName",  
    listenerObjectORFunction);
```

- Se pueden registrar varios detectores en una instancia de componente.

```
myButton.addEventListener("click", listener1);  
myButton.addEventListener("click", listener2);
```

- Se puede registrar un detector en varias instancias de componente.

```
myButton.addEventListener("click", listener1);  
myButton2.addEventListener("click", listener1);
```

- Se pasa a la función de controlador un objeto de evento.

Puede utilizar el objeto de evento en el cuerpo de la función para recuperar información sobre el tipo de evento y la instancia que difunde el evento. Véase [“El objeto de evento” en la página 82](#).

- Un objeto detector permanece activo hasta que se elimina explícitamente mediante `EventDispatcher.removeEventListener()`. Por ejemplo:

```
myComponent.removeEventListener("change", listenerObj);
```

Utilización de objetos detectores

Para utilizar un objeto detector, puede usar la palabra clave `this` para especificar el objeto actual como el detector, usar un objeto que ya existe en la aplicación o crear un objeto nuevo.

- Use `this` en la mayoría de las situaciones.
Suele ser más sencillo usar el objeto actual (`this`) como un detector, ya que su ámbito contiene los componentes que deben reaccionar cuando se difunde el evento.
- Use un objeto existente si es conveniente.
Por ejemplo, en una aplicación de formularios Flash, puede interesarle usar un formulario como objeto detector si dicho formulario contiene los componentes que reaccionan al evento. Coloque el código en un fotograma de la línea de tiempo del formulario.
- Use un nuevo objeto detector si muchos componentes están difundiendo un evento (por ejemplo, el evento `click`) y sólo desea que respondan determinados objetos detectores.

Si utiliza el objeto `this`, defina una función con el mismo nombre que el del evento que desea gestionar; la sintaxis es la siguiente:

```
function eventName(evtObj:Object){  
    // escribir código aquí  
};
```

Si desea usar un nuevo objeto detector, debe crear el objeto, definir una propiedad con el mismo nombre que los eventos y asignar la propiedad a una función callback que se ejecuta cuando se difunde el evento, de la manera siguiente:

```
var listenerObject:Object = new Object();  
listenerObject.eventName = function(evtObj:Object){  
    // escribir código aquí  
};
```

Si desea usar un objeto existente, utilice la misma sintaxis que un objeto detector nuevo, sin crear el objeto nuevo, como se indica a continuación:

```
existingObject.eventName = function(evtObj:Object){  
    // escribir código aquí  
};
```

SUGERENCIA

El parámetro `evtObj` es un objeto que se genera automáticamente cuando se activa un evento y se pasa a la función callback. El objeto de evento tiene propiedades que contienen información sobre el evento. Para ver más detalles, consulte [“El objeto de evento” en la página 82](#).

Por último, llame al método `addEventListener()` desde la instancia del componente que difunde el evento. El método `addEventListener()` usa dos parámetros: una cadena que indica el nombre del evento y una referencia al objeto detector.

```
componentInstance.addEventListener("eventName", listenerObject);
```

Éste es el segmento de código completo, que puede copiar y pegar. Asegúrese de reemplazar el código en cursiva por los valores reales; puede utilizar `listenerObject` y `evtObj` o cualquier otro identificador válido, pero debe cambiar `eventName` por el nombre del evento.

```
var listenerObject:Object = new Object();  
listenerObject.eventName = function(evtObj:Object){  
    // el código que añada aquí se ejecutará  
    // cuando se active el evento  
};  
componentInstance.addEventListener("eventName", listenerObject);
```

En el siguiente segmento de código se usa la palabra clave `this` como objeto detector:

```
function eventName(evtObj:Object){  
    // el código que añada aquí se ejecutará  
    // cuando se active el evento  
}  
componentInstance.addEventListener("eventName", this);
```

Puede llamar a `addEventListener()` desde cualquier instancia de componente; se añade a cada componente desde la clase `EventDispatcher`. (Una adición es una clase que proporciona características específicas que amplían el comportamiento de otra clase.) Para más información, consulte “`EventDispatcher.addEventListener()`” en *Referencia del lenguaje de componentes*.

Para más información sobre los eventos que difunde un componente, consulte la entrada correspondiente de cada componente en *Referencia del lenguaje de componentes*. Por ejemplo, los eventos del componente `Button` se muestran en la sección del componente `Button` (o en `Ayuda > Referencia del lenguaje de componentes > Componente Button > Clase Button > Resumen de eventos de la clase Button`).

Para registrar un objeto detector en un archivo de Flash (FLA):

1. En Flash, seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente Button desde el panel Componentes al escenario.
3. En el inspector de propiedades, introduzca el nombre de instancia **myButton**.
4. Arrastre un componente TextInput desde el panel Componentes al escenario.
5. En el inspector de propiedades, introduzca el nombre de instancia **myText**.
6. Seleccione el fotograma 1 de la línea de tiempo.
7. Seleccione Ventana > Acciones.
8. En el panel Acciones, introduzca el siguiente código:

```
var myButton:mx.controls.Button;  
var myText:mx.controls.TextInput;
```

```
function click(evt){  
    myText.text = evt.target;  
}
```

```
myButton.addEventListener("click", this);
```

La propiedad `target` del objeto de evento, `evt`, es una referencia a la instancia que difunde el evento. Este código muestra el valor de la propiedad `target` en el componente `TextInput`.

Para registrar un objeto detector en un archivo de clase (AS):

1. Abra el archivo `TipCalculator.fla` desde la ubicación especificada en [“Trabajo con componentes” en la página 53](#).
2. Abra el archivo `TipCalculator.as` desde la ubicación especificada en [“Trabajo con componentes” en la página 53](#).
3. En el archivo FLA, seleccione `form1` y vea el nombre de la clase, `TipCalculator`, en el inspector de propiedades.

Éste es el vínculo entre el formulario y el archivo de clase. Todo el código para esta aplicación está en el archivo `TipCalculator.as`. El formulario incorpora las propiedades y los comportamientos definidos por la clase que se le ha asignado.

4. En el archivo AS, desplácese a la línea 25, `public function onLoad():Void`.

La función `onLoad()` se ejecuta cuando el formulario se carga en Flash Player. En el cuerpo de la función, la instancia `subtotal` de `TextInput` y las tres instancias de `RadioButton`, `percentRadio15`, `percentRadio18` y `percentRadio20`, llaman al método `addEventListener()` para registrar un detector con un evento.

5. Vea la línea 27, `subtotal.addEventListener("change", this)`.
Al llamar a `addEventListener()`, debe pasarle dos parámetros. El primero es una cadena que indica el nombre del evento que se difunde (en este caso, "change"). El segundo es una referencia a un objeto o una función que gestiona el evento. En este caso, el parámetro es la palabra clave `this`, que hace referencia a una instancia del archivo de clase (un objeto). A continuación, Flash busca en el objeto una función con el nombre del evento.
6. Vea la línea 63, `public function change(event:Object):Void`.
Ésta es la función que se ejecuta cuando cambia la instancia `subtotal` de `TextInput`.
7. Seleccione el archivo `TipCalculator.fla` y seleccione `Control > Probar película` para probar el archivo.

Utilización de la función callback `handleEvent`

También puede utilizar objetos detectores que admitan la función `handleEvent`. Independientemente del nombre del evento que se difunde, se llama al método `handleEvent` del objeto detector. Para gestionar varios eventos, debe utilizar una sentencia `if..else` o una sentencia `switch`. Por ejemplo, el código siguiente utiliza una sentencia `if..else` para gestionar los eventos `click` y `change`:

```
// definir la función handleEvent
// pasar evt como parámetro del objeto de evento

function handleEvent(evt){
    // comprobar si el evento fue click
    if (evt.type == "click"){
        // hacer algo si el evento fue click
    } else if (evt.type == "change"){
        // hacer otra cosa si el evento fue change
    }
}

// registrar el objeto detector para
// dos instancias de componente distintas
// como la función está definida en
// el objeto "this", el detector es this.

instance.addEventListener("click", this);
instance2.addEventListener("change", this);
```

Utilización de funciones de detector

A diferencia de la sintaxis de `handleEvent`, varias funciones de detector pueden gestionar distintos eventos. Así, en lugar de usar las comprobaciones `if` y `else` en `myHandler`, puede definir simplemente `myChangeListener` para el evento `change` y `myScrollHandler` para el evento `scroll`, y registrarlos de la manera siguiente:

```
myList.addEventListener("change", myChangeListener);
myList.addEventListener("scroll", myScrollHandler);
```

Para utilizar una función de detector, primero debe definir una función:

```
function myFunction:Function(evtObj:Object){
    // escribir código aquí
}
```

SUGERENCIA

El parámetro `evtObj` es un objeto que se genera automáticamente cuando se activa un evento y se pasa a la función. El objeto de evento tiene propiedades que contienen información sobre el evento. Para ver más detalles, consulte [“El objeto de evento” en la página 82](#).

A continuación, se llama al método `addEventListener()` desde la instancia del componente que difunde el evento. El método `addEventListener()` usa dos parámetros: una cadena que indica el nombre del evento y una referencia a la función.

```
componentInstance.addEventListener("eventName", myFunction);
```

Puede llamar a `addEventListener()` desde cualquier instancia de componente; se incluye en cada componente de interfaz de usuario desde la clase `EventDispatcher`. Para más información, consulte [EventDispatcher.addEventListener\(\)](#).

Para más información sobre los eventos que difunde un componente, consulte la entrada correspondiente de cada componente en *Referencia del lenguaje de componentes*.

Para registrar un objeto detector en un archivo de Flash (FLA):

1. En Flash, seleccione Archivo > Nuevo y cree un documento de Flash.
2. Arrastre un componente List desde el panel Componentes al escenario.
3. En el inspector de propiedades, introduzca el nombre de la instancia `myList`.
4. Seleccione el fotograma 1 de la línea de tiempo.
5. Seleccione Ventana > Acciones.

6. En el panel Acciones, introduzca el siguiente código:

```
// declarar variables
var myList:mx.controls.List;
var myHandler:Function;

// añadir elementos a la lista
myList.addItem("Bird");
myList.addItem("Dog");
myList.addItem("Fish");
myList.addItem("Cat");
myList.addItem("Ape");
myList.addItem("Monkey");

// definir función myHandler
function myHandler(eventObj:Object){

    // usar el parámetro eventObj
    // para capturar el tipo de evento
    if (eventObj.type == "change"){
        trace("The list changed");
    } else if (eventObj.type == "scroll"){
        trace("The list was scrolled");
    }
}

// Registrar la función myHandler con myList.
// Cuando se selecciona un elemento (se activa el evento change) o
// se recorre la lista, se ejecuta myHandler.
myList.addEventListener("change", myHandler);
myList.addEventListener("scroll", myHandler);
```

NOTA

La propiedad `type` del objeto de evento, `evt`, es una referencia al nombre de evento.

7. Seleccione Control > Probar película; a continuación, seleccione un elemento de la lista y recorra la lista para ver los resultados en el panel Salida.

ATENCIÓN

En una función de detector, la palabra clave `this` hace referencia a la instancia de componente que llama a `addEventListener()`, no a la línea de tiempo o la clase en la que está definida la función. No obstante, puede utilizar la clase `Delegate` para delegar la función de detector en un ámbito diferente. Véase [“Delegación de eventos” en la página 78](#). Para ver un ejemplo de ámbito de la función, consulte la siguiente sección.

Ámbito en detectores

Ámbito hace referencia al objeto en el que se ejecuta una función. Todas las referencias de variable de la función se reconocen como propiedades de ese objeto. Puede utilizar la clase Delegate para especificar el ámbito de un detector. Para más información, consulte [“Delegación de eventos” en la página 78](#).

Como ya se indicó, se registra un detector con una instancia de componente llamando a `addEventListener()`. Este método usa dos parámetros: una cadena que indica el nombre del evento y una referencia a un objeto o una función. En la tabla siguiente se muestra el ámbito de cada tipo de parámetro:

Tipo de detector	Ámbito
Objeto	Objeto detector.
Función	Instancia de componente que difunde el evento.

Si pasa un objeto `addEventListener()`, la función callback asignada a dicho objeto (o la función definida en ese objeto) se invoca en el ámbito del objeto. Esto significa que la palabra clave `this`, cuando se usa dentro de la función callback, hace referencia al objeto detector de la manera siguiente:

```
var lo:Object = new Object();
lo.click = function(evt){
    // this hace referencia al objeto lo
    trace(this);
}
myButton.addEventListener("click", lo);
```

Sin embargo, si pasa una función a `addEventListener()`, la función se invoca en el ámbito de la instancia de componente que llama a `addEventListener()`. Esto significa que la palabra clave `this`, cuando se usa en la función, hace referencia a la instancia de componente que difunde el evento. Esto supone un problema si se está definiendo la función en un archivo de clase. No puede acceder a las propiedades y métodos del archivo de clase con las rutas esperadas porque `this` no señala a una instancia de la clase. Para solucionar este problema, use la clase Delegate para delegar una función en el ámbito correcto. Véase [“Delegación de eventos” en la página 78](#).

El código siguiente ilustra el ámbito de una función cuando se pasa a `addEventListener()` en un archivo de clase. Para utilizar este código, cópielo en un archivo de ActionScript (AS) denominado `Cart.as`. Cree un archivo de Flash (FLA) con un componente `Button`, `myButton`, y un componente `DataGrid`, `myGrid`. Seleccione ambos componentes en el escenario y presione F8 para convertirlos en un símbolo nuevo denominado `Cart`. En las Propiedades de vinculación del símbolo de `Cart`, asígnele la clase `Cart`.

```

class Cart extends MovieClip {

    var myButton:mx.controls.Button;
    var myGrid:mx.controls.DataGrid;

    function myHandler(eventObj:Object){

        // Usar el parámetro eventObj
        // para capturar el tipo de evento.
        if (eventObj.type == "click"){

            /* Enviar el valor de this al panel Salida.
            Como myHandler es una función que no está definida
            en un objeto detector, this es una referencia a la
            instancia de componente para la que myHandler está registrada
            (myButton). Además, como this no hace referencia a una
            instancia de la clase Cart, myGrid está sin definir.
            */
            trace("this: " + this);
            trace("myGrid: " + myGrid);
        }
    }

    // registrar la función myHandler con myButton
    // cuando se hace clic en el botón, se ejecuta myHandler

    function onLoad():Void{
        myButton.addEventListener("click", myHandler);
    }
}

```

Delegación de eventos

Puede importar la clase Delegate en sus scripts o clases para delegar eventos en funciones y ámbitos específicos (consulte “Clase Delegate” en *Referencia del lenguaje de componentes*). Para importar la clase Delegate, 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 al parámetro *function* especificado.

Hay dos usos comunes para llamar a `Delegate.create()`:

- Para distribuir el mismo evento a dos funciones distintas.
Véase el apartado siguiente.

- Para llamar a funciones dentro del ámbito de la clase contenedora.

Cuando se pasa una función como un parámetro a `addEventListener()`, la función se invoca en el ámbito de la instancia del componente difusor, no el del objeto en el que está declarada. Véase [“Delegación del ámbito de una función” en la página 81](#).

Delegación de eventos a funciones

Llamar a `Delegate.create()` resulta útil cuando se tienen dos componentes que difunden eventos del mismo nombre. Por ejemplo, si se tiene una casilla de verificación y un botón, se tendría que usar la instrucción `switch` en los datos que se obtengan de la propiedad `eventObject.target` para determinar qué componente está difundiendo el evento `click`.

Para utilizar el código siguiente, coloque una casilla de verificación denominada `myCheckBox_chb` y un botón denominado `myButton_btn` en el escenario. Seleccione ambas instancias y presione F8 para crear un nuevo símbolo. Haga clic en Avanzado si el cuadro de diálogo está en modo básico y seleccione Exportar para ActionScript. Introduzca **Cart** en el cuadro de texto Clase de AS 2.0. En el inspector de propiedades, asigne al nuevo símbolo el nombre de instancia que desee. El símbolo quedará asociado con la clase `Cart` y una instancia del símbolo se convertirá en una instancia de esta clase.

```
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    function onLoad() {
        myCheckBox_chb.addEventListener("click", this);
        myButton_btn.addEventListener("click", this);
    }

    function click(eventObj:Object) {
        switch(eventObj.target) {
            case myButton_btn:
                // envía el nombre de la instancia de difusor
                // y el tipo de evento al panel Salida
                trace(eventObj.target + ": " + eventObj.type);
                break;
            case myCheckBox_chb:
                trace(eventObj.target + ": " + eventObj.type);
        }
    }
}
```

```

        break;
    }
}

```

El siguiente código muestra el mismo archivo de clase (Cart.as) modificado para usar Delegate:

```

import mx.utils.Delegate;
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    function onLoad() {
        myCheckBox_chb.addEventListener("click", Delegate.create(this,
        chb_onClick));
        myButton_btn.addEventListener("click", Delegate.create(this,
        btn_onClick));
    }

    // dos funciones independientes gestionan los eventos

    function chb_onClick(eventObj:Object) {
        // envía el nombre de la instancia de difusor
        // y el tipo de evento al panel Salida
        trace(eventObj.target + ": " + eventObj.type);
        // envía la ruta absoluta del símbolo
        // que asoció con la clase Cart
        // del archivo FLA al panel Salida
        trace(this)
    }

    function btn_onClick(eventObj:Object) {
        trace(eventObj.target + ": " + eventObj.type);
    }
}

```


Delegación del ámbito de una función

El método `addEventListener()` requiere dos parámetros: el nombre de un evento y una referencia a un detector. El detector puede ser un objeto o una función. Si pasa un objeto, la función callback asignada al objeto se invoca en el ámbito del objeto. No obstante, si pasa una función, la función se invoca en el ámbito de la instancia de componente que llama a `addEventListener()`. Para más información, consulte [“Ámbito en detectores” en la página 77](#).

Como la función se invoca en el ámbito de la instancia del difusor, la palabra clave `this` del cuerpo de la función señala la instancia de difusor, no la clase que contiene la función. Por tanto, no puede acceder a las propiedades y los métodos de la clase que contiene la función. Use la clase `Delegate` para delegar el ámbito de una función en la clase contenedora, a fin de poder acceder a las propiedades y los métodos de la clase contenedora.

En el ejemplo siguiente se usa el mismo enfoque que en la sección anterior con una variación del archivo de clase `Cart.as`:

```
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {

    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    // definir una variable para acceder
    // desde la función chb_onClick
    var i:Number = 10

    function onLoad() {
        myCheckBox_chb.addEventListener("click", chb_onClick);
    }

    function chb_onClick(eventObj:Object) {
        // Esperaría poder acceder
        // a la variable i, y obtener 10 como resultado.
        // Sin embargo, this envía undefined
        // al panel Salida porque
        // la función no está en el ámbito de
        // la instancia de Cart en la que está definida i.
        trace(i);
    }
}
```

Para acceder a las propiedades y los métodos de la clase `Cart`, llame a `Delegate.create()` como segundo parámetro de `addEventListener()`, de la manera siguiente:

```
import mx.utils.Delegate;
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;
    // definir una variable para acceder
    // desde la función chb_onClick
    var i:Number = 10

    function onLoad() {
        myCheckBox_chb.addEventListener("click", Delegate.create(this,
        chb_onClick));
    }

    function chb_onClick(eventObj:Object) {
        // Envía 10 al panel Salida
        // porque la función está dentro del ámbito de
        // la instancia de Cart
        trace(i);
    }
}
```

El objeto de evento

El objeto de evento es una instancia de la clase `Object` de `ActionScript`; tiene las siguientes propiedades que contienen información sobre un evento.

Propiedad	Descripción
<code>type</code>	Cadena que indica el nombre del evento.
<code>target</code>	Referencia a la instancia del componente que difunde el evento.

Cuando un evento tiene propiedades adicionales, se muestran en la entrada del evento en el Diccionario de componentes.

El objeto de evento se genera automáticamente cuando se activa un evento y se pasa a la función `callback` del objeto detector o a la función de detector.

El objeto de evento se puede utilizar dentro de la función para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento. A partir del nombre de la instancia, puede acceder a otras propiedades de componente. 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:

```
var myButton:mx.controls.Button;
var listener:Object;

listener = new Object();

listener.click = function(evtObj){
    trace("The " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Utilización del controlador de eventos `on()`

El controlador de eventos `on()` puede asignarse a una instancia de componente igual que se asigna un controlador a un botón o a un clip de película. Un controlador de eventos `on()` puede ser útil para realizar pruebas sencillas pero es preferible utilizar detectores de eventos para todas las aplicaciones. Para más información, consulte [“Utilización de detectores para gestionar eventos” en la página 70](#).

Cuando utilice la palabra clave `this` en un controlador `on()` directamente asociado a un componente (asignado a la *instancia* del componente en el panel Acciones), `this` hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado directamente a la instancia `myButton` del componente `Button`, muestra “`_level0.myButton`” en el panel Salida:

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

Para utilizar el controlador de eventos `on()`:

1. Arrastre un componente de interfaz de usuario al escenario.
Por ejemplo, arrastre un componente `Button` al escenario.
2. Seleccione el componente en el escenario y abra el panel Acciones.

3. Añada el controlador `on()` al panel Acciones con el formato:

```
on(event){  
    //your statements go here  
}
```

Por ejemplo:

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

Flash ejecuta el código del controlador `on()` cuando se produce el evento del controlador `on()` (en este caso, un clic de botón).

4. Seleccione Control > Probar película y haga clic en el botón para ver el resultado.

Personalización de componentes

Puede cambiar la apariencia de los componentes cuando los utiliza en diferentes aplicaciones. Para personalizar la apariencia de los componentes, utilice los tres elementos siguientes, de forma individual o conjunta:

Estilos Los componentes de interfaz de usuario (IU) tienen propiedades de estilo que definen la apariencia de algunos aspectos de un componente. Cada componente tiene su propio conjunto de propiedades de estilo modificables y no todos los aspectos visuales de un componente pueden cambiarse mediante la definición de un estilo. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 86](#).

Aspectos Un *aspecto* contiene una colección de símbolos que forman la visualización gráfica de un componente. La *aplicación de aspectos* es el proceso mediante el cual se cambia la apariencia de un componente, modificando o sustituyendo sus gráficos de origen. Un aspecto puede ser una parte pequeña, como un borde o esquina de un borde, o bien una parte compuesta como la imagen completa de un botón sin oprimir (estado en el que se encuentra cuando no se ha presionado). Asimismo, puede ser un símbolo sin gráfico que contiene código que dibuja una parte del componente. Algunos aspectos de un componente que no pueden establecerse a través de sus propiedades de estilo, sí pueden definirse mediante la modificación del aspecto. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 102](#).

Temas Un tema es una colección de estilos y aspectos que puede guardar como un archivo FLA y aplicar a otro documento. Para más información, consulte [“Temas” en la página 115](#).

Este capítulo contiene las siguientes secciones:

Utilización de estilos para personalizar el texto y el color de un componente . . .	86
Aplicación de aspectos a los componentes	102
Temas	115
Combinación de aplicación de aspectos y estilos para personalizar un componente	126

Utilización de estilos para personalizar el texto y el color de un componente

Flash proporciona propiedades de estilo que pueden editarse para cada componente de interfaz de usuario. La documentación de cada componente específico contiene una tabla donde se enumeran los estilos modificables de dicho componente (por ejemplo, puede ver una tabla de estilos del componente Accordion en “Utilización de estilos con el componente Accordion” en *Referencia del lenguaje de componentes*). Además, los componentes de interfaz de usuario heredan los métodos `setStyle()` y `getStyle()` de la clase `UIObject` (véase `UIObject.setStyle()` y `UIObject.getStyle()`). Puede utilizar los métodos `setStyle()` y `getStyle()` en una instancia de componente para establecer y obtener los valores de las propiedades de estilo, tal y como se explica posteriormente en “Definición de los estilos en una instancia de un componente” en la página 89.

NOTA

No puede establecer estilos para los componentes multimedia.

Utilización de declaraciones de estilo y temas

En un ámbito más amplio, los estilos se organizan en *declaraciones de estilo*, que permiten controlar los valores de las propiedades de estilo en varias instancias de componentes. Una declaración de estilo es un objeto que crea la clase `CSSStyleDeclaration` y cuyas propiedades son las configuraciones de estilo que pueden asignarse a los componentes. Las declaraciones de estilo en ActionScript se modelan según la forma en que las hojas de estilos en cascada (CSS) afectan a las páginas HTML. En las páginas HTML, puede crear un archivo de hoja de estilos que defina las propiedades de estilo relativas al contenido de un grupo de páginas HTML. Con los componentes, puede crear un objeto de declaración de estilo y añadirle propiedades de estilo para controlar la apariencia de los componentes de un documento de Flash.

Además, las declaraciones de estilo se organizan en *temas*. Flash proporciona dos temas visuales para componentes: Halo (HaloTheme.fla) y Sample (SampleTheme.fla). Un *tema* es un conjunto de estilos y gráficos que controla la apariencia de los componentes en un documento. Cada tema proporciona estilos a los componentes. Los estilos usados por cada componente dependen en parte del tema que use el documento. Algunos estilos, como `defaultIcon`, se usan en los componentes asociados independientemente del tema aplicado al documento. Otros estilos, como `themeColor` y `symbolBackgroundColor`, sólo se usan en componentes si se está usando el tema correspondiente. Por ejemplo, `themeColor` sólo se usa si se está usando el tema Halo y `symbolBackgroundColor` sólo se usa si se está usando el tema Sample. Para saber qué propiedades de estilo puede establecer para un componente, debe saber qué tema se asigna a dicho componente. Las tablas de estilos para cada componente de *Referencia del lenguaje de componentes* indican si cada propiedad de estilo se aplica a uno de los temas suministrados (o a los dos). Para más información, consulte [“Temas” en la página 115](#).

Aspectos básicos de la configuración de estilos

Cuando utilice estilos y declaraciones de estilo, observará que puede establecer estilos de diversas formas: globalmente, en el tema, en la declaración de estilo o en la propiedad de estilo. Además, algunas propiedades de estilo pueden heredarse de un componente principal (por ejemplo, un panel secundario Accordion puede heredar el tratamiento de fuentes del componente Accordion). A continuación se destacan algunos puntos clave del comportamiento de estilos:

Dependencia del tema El tema actual determina las propiedades de estilo que pueden establecerse en un componente concreto. De forma predeterminada, los componentes de Flash se han diseñado para utilizar el tema Halo, pero Flash proporciona además un tema Sample. Así pues, cuando lea una tabla de propiedades de estilo, como la correspondiente al componente Button que encontrará en *“Utilización de estilos con el componente Button”* en *Referencia del lenguaje de componentes*, observe qué tema es compatible con el estilo que desea. En la tabla se indica Halo, Sample o Ambos (que significa que ambos temas admiten la propiedad de estilo). Para cambiar el tema actual, consulte [“Cambio entre temas” en la página 116](#).

Herencia No es posible definir la herencia en el código ActionScript. Un elemento secundario del componente se diseña para heredar o no un estilo del componente principal.

Hojas de estilos globales Las declaraciones de estilo en Flash no admiten la estructura “en cascada” en los documentos de Flash de la forma en que se utilizan las hojas de estilos en cascada en los documentos HTML. Todos los objetos de declaración de hojas de estilos se definen globalmente en toda la aplicación.

Prioridad Si un estilo de componente se define de más de una forma (por ejemplo, si se establece `textColor` globalmente y en el nivel de instancia del componente), Flash utiliza el primer estilo que encuentra, según el orden definido en [“Utilización de un estilo global y estilos personalizados y de clase en un mismo documento”](#) en la página 97.

Definición de estilos

La existencia de las propiedades de estilo, su organización en las declaraciones de estilo y la organización más amplia de las declaraciones de estilo y los gráficos en temas permite personalizar un componente de las siguientes formas:

- Definir los estilos en una instancia de componente.
Puede cambiar las propiedades de texto y de color de una única instancia de componente. Este recurso es efectivo en determinadas situaciones, aunque puede tardar mucho tiempo si necesita definir propiedades individuales en todos los componentes de un documento.
Para más información, consulte [“Definición de los estilos en una instancia de un componente”](#) en la página 89.
- Ajustar la declaración de estilo global que define los estilos de todos los componentes de un documento.
Si desea que el documento completo tenga un aspecto uniforme, puede crear estilos en la declaración de estilo global.
Para más información, consulte [“Definición de estilos globales”](#) en la página 91.
- Crear declaraciones de estilo personalizado y aplicarlas a instancias de componente específicas.
Puede hacer que grupos de componentes de un documento compartan un mismo estilo. Para ello, puede crear declaraciones de estilo personalizado para aplicarlas a componentes específicos.
Para más información, consulte [“Definición de estilos personalizados para grupos de componentes”](#) en la página 92.
- Crear declaraciones de estilo de clase predeterminado.
Puede definir una declaración de estilo de clase predeterminado para que todas las instancias de una misma clase compartan una apariencia predeterminada.
Para más información, consulte [“Definición de los estilos de una clase de componente”](#) en la página 94.

- Usar estilos heredados para establecer estilos de componentes en una parte de un documento.

Los componentes añadidos a un contenedor heredan los valores de las propiedades de estilo establecidas para el contenedor.

Para más información, consulte [“Definición de estilos heredados en un contenedor” en la página 95](#).

Flash no muestra los cambios realizados en las propiedades de estilo al visualizar componentes en el escenario mediante la característica Previsualización dinámica. Para más información, consulte [“Componentes en previsualización dinámica” en la página 65](#).

Definición de los estilos en una instancia de un componente

Puede escribir código ActionScript para definir y obtener las propiedades de estilo de cualquier instancia de componente. Los métodos `UIObject.setStyle()` y `UIObject.getStyle()` pueden llamarse directamente desde cualquier componente de interfaz de usuario. La sintaxis siguiente especifica una propiedad y un valor para una instancia de componente:

```
instanceName.setStyle("propertyName", value);
```

Por ejemplo, el código siguiente establece los colores de realce en una instancia de `Button` denominada `myButton` que utiliza el tema `Halo`:

```
myButton.setStyle("themeColor", "haloBlue");
```

NOTA

Si el valor es una cadena, debe ir entrecomillado.

También es posible acceder a los estilos directamente como propiedades (por ejemplo, `myButton.color = 0xFF00FF`).

Las propiedades de estilo establecidas en una instancia de componente mediante `setStyle()` tienen la prioridad más alta y reemplazan las demás configuraciones de estilo basadas en la declaración de estilo o en el tema. Sin embargo, cuanto más propiedades se definan mediante `setStyle()` en una sola instancia de componente, más lenta será la representación del componente en tiempo de ejecución. Para acelerar la representación de un componente personalizado mediante código `ActionScript` que defina las propiedades de estilo durante la creación de la instancia de componente, utilice `UIObject.createClassObject()`, descrito en *Referencia del lenguaje de componentes*, e incluya la configuración de estilo en el parámetro `initObject`. Por ejemplo, con un componente `ComboBox` en la biblioteca de documentos actual, el siguiente código crea una instancia de `ComboBox` denominada `my_cb` y establece el texto del cuadro combinado como texto en cursiva y alineado a la derecha:

```
createClassObject(mx.controls.ComboBox, "my_cb", 1, {fontStyle:"italic",
    textAlign:"right"});
my_cb.addItem({data:1, label:"One"});
```

NOTA

Si desea cambiar varias propiedades o cambiar propiedades de varias instancias de componente, debe crear un estilo personalizado. Una instancia de componente que utilice un estilo personalizado para varias propiedades se representará con mayor rapidez que una instancia de componente donde se realicen diversas llamadas a `setStyle()`. Para más información, consulte [“Definición de estilos personalizados para grupos de componentes” en la página 92](#).

Para establecer o cambiar una propiedad para una instancia de componente individual que utiliza el tema Halo:

1. Seleccione la instancia de componente en el escenario.
2. En el inspector de propiedades, asígnele el nombre de instancia `myComponent`.
3. Abra el panel Acciones, seleccione Escena 1 y, a continuación, seleccione Capa 1: Fotograma 1.

4. Escriba el código siguiente para cambiar el color de la instancia a naranja:

```
myComponent.setStyle("themeColor", "haloOrange");
```

5. Seleccione Control > Probar película para ver los cambios.

Para ver una lista de estilos admitidos por un componente específico, consulte la entrada del componente en *Referencia del lenguaje de componentes*.

Para crear una instancia de componente y establecer varias propiedades simultáneamente mediante código ActionScript:

1. Arrastre un componente a la biblioteca.
2. Abra el panel Acciones, seleccione Escena 1 y, a continuación, seleccione Capa 1: Fotograma 1.
3. Escriba la siguiente sintaxis para crear una instancia del componente y establecer sus propiedades:

```
createClassObject(className, "instance_name", depth, {style:"setting", style:"setting"});
```

De esta forma, por ejemplo, con un componente Button en la biblioteca, el siguiente código ActionScript crea una instancia de botón `my_button` en la profundidad 1 con los estilos de texto establecidos en color púrpura y cursiva:

```
createClassObject(mx.controls.Button, "my_button", 1, {label:"Hello", color:"0x9900CC", fontStyle:"italic"});
```

Para más información, consulte `UIObject.createClassObject()`.

4. Seleccione Control > Probar película para ver los cambios.

Para ver una lista de estilos admitidos por un componente específico, consulte la entrada del componente en *Referencia del lenguaje de componentes*.

Definición de estilos globales

De forma predeterminada, todos los componentes siguen una declaración de estilo global hasta que se asocia al componente otra declaración de estilo (véase [“Definición de estilos personalizados para grupos de componentes” en la página 92](#)). La declaración de estilo global se asigna a todos los componentes de Flash creados con la versión 2 de la arquitectura de componentes de Macromedia. El objeto `_global` tiene una propiedad denominada `style` (`_global.style`) que es una instancia de `CSSStyleDeclaration` y que actúa como la declaración de estilo global. Si cambia el valor de una propiedad de estilo en la declaración de estilo global, el cambio se aplica a todos los componentes del documento de Flash.

ATENCIÓN

Algunos estilos se definen en la instancia `CSSStyleDeclaration` de la clase de componente (por ejemplo, el estilo `backgroundColor` de los componentes `TextArea` y `TextInput`). Como la declaración de estilo de la clase tiene prioridad sobre la declaración de estilo global cuando se determinan los valores de estilo, establecer el valor de `backgroundColor` en la declaración de estilo global no surtirá efecto en los componentes `TextArea` y `TextInput`. Para más información sobre la prioridad de estilos, consulte [“Utilización de un estilo global y estilos personalizados y de clase en un mismo documento” en la página 97](#). Para más información sobre la edición de la instancia `CSSStyleDeclaration` de una clase de componente, consulte [“Definición de los estilos de una clase de componente” en la página 94](#).

Para cambiar una o más propiedades en la declaración de estilo global:

1. Asegúrese de que el documento contiene como mínimo una instancia de componente. Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 54](#).
2. Seleccione en la línea de tiempo el fotograma en el que aparecen los componentes (o el fotograma anterior).
3. En el panel Acciones, use código como el siguiente para cambiar los valores de las propiedades en la declaración de estilo global. Sólo tiene que enumerar las propiedades cuyos valores desea modificar, como se indica a continuación:

```
_global.style.setStyle("color", 0xCC6699);  
_global.style.setStyle("themeColor", "haloBlue")  
_global.style.setStyle("fontSize",16);  
_global.style.setStyle("fontFamily" , "_serif");
```

4. Seleccione Control > Probar película para ver los cambios.

Definición de estilos personalizados para grupos de componentes

Puede crear declaraciones de estilo personalizado para especificar un conjunto exclusivo de propiedades para grupos de componentes de un documento de Flash. Además de la propiedad `style` del objeto `_global` (descrita en [“Definición de estilos globales” en la página 91](#)), que determina la declaración de estilo predeterminado para todo un documento de Flash, el objeto `_global` tiene también una propiedad `styles`, que es una lista de declaraciones de estilo personalizado disponibles. Así pues, puede crear una declaración de estilo como una nueva instancia del objeto `CSSStyleDeclaration`, asignarle un nombre de estilo personalizado e incluirla en la lista `_global.styles`. A continuación, puede especificar las propiedades y valores del estilo, y asignar el nombre de estilo a las instancias del componente que deben compartir el mismo aspecto.

Recuerde que cuando asigna el nombre de estilo a una instancia de componente, éste sólo responde a las propiedades de estilo que admita el componente. Para ver una lista de las propiedades de estilo admitidas por cada componente, consulte las entradas individuales de componente en *Referencia del lenguaje de componentes*.

Para realizar cambios en un formato de estilo personalizado, use la sintaxis siguiente:

```
_global.styles.CustomStyleName.setStyle(propertyName, propertyValue);
```

Las configuraciones de estilo personalizado tienen prioridad sobre las configuraciones de estilo de clase, heredado o global. Para ver una lista de la prioridad de estilos, consulte [“Utilización de un estilo global y estilos personalizados y de clase en un mismo documento” en la página 97.](#)

Para crear una declaración de estilo personalizado para un grupo de componentes:

1. Añada, al menos, un componente al escenario.

Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 54.](#)

En este ejemplo se utilizan tres componentes de botón con los nombres de instancia a, b y c. Si utiliza otros componentes, asígneles nombres de instancia en el inspector de propiedades y utilice dichos nombres en el paso 8.

2. Seleccione en la línea de tiempo el fotograma en el que aparece el componente (o el fotograma posterior).
3. Abra el panel Acciones.

4. Añada la siguiente sentencia `import` para tener acceso a la función constructora y poder crear una nueva declaración de estilo desde la clase `CSSStyleDeclaration`:

```
import mx.styles.CSSStyleDeclaration;
```

5. Utilice la sintaxis siguiente para crear una instancia del objeto `CSSStyleDeclaration` con el fin de definir el nuevo formato de estilo personalizado:

```
var new_style:Object = new CSSStyleDeclaration();
```

6. Asigne a la declaración de estilo un nombre como, por ejemplo, `myStyle`, en la lista `_global.styles` de declaraciones de estilo personalizado e identifique el objeto que contiene todas las propiedades de la nueva declaración de estilo.

```
_global.styles.myStyle = new_style;
```

7. Utilice el método `setStyle()` (heredado de la clase `UIObject`) para añadir al objeto `new_style` propiedades que se asocian a su vez con la declaración de estilo personalizado `myStyle`:

```
new_style.setStyle("fontFamily", "_serif");  
new_style.setStyle("fontSize", 14);  
new_style.setStyle("fontWeight", "bold");  
new_style.setStyle("textDecoration", "underline");  
new_style.setStyle("color", 0x666699);
```

8. En el mismo panel Script, utilice la sintaxis siguiente para definir la propiedad `styleName` de tres componentes específicos en el nombre de la declaración de estilo personalizado:

```
a.setStyle("styleName", "myStyle");
b.setStyle("styleName", "myStyle");
c.setStyle("styleName", "myStyle");
```

Ahora *puede* acceder a los estilos de la declaración de estilo personalizado mediante los métodos `setStyle()` y `getStyle()`, a través de la propiedad `styleName` global de la declaración. Por ejemplo, el código siguiente define el estilo `backgroundColor` en la declaración de estilo `myStyle`:

```
_global.styles.myStyle.setStyle("themeColor", "haloOrange");
```

Sin embargo, en los pasos 5 y 6 se asoció la instancia `new_style` con la declaración de estilo, de forma que es posible utilizar la sintaxis más corta,

```
comonew_style.setStyle("themeColor", "haloOrange").
```

Para más información sobre los métodos `setStyle()` y `getStyle()`, consulte

`UIObject.setStyle()` y `UIObject.getStyle()`.

Definición de los estilos de una clase de componente

Puede definir una declaración de estilo de clase para cualquier clase de componente (Button, CheckBox, etc.) que defina los estilos predeterminados para cada instancia de dicha clase.

Antes de crear las instancias, es preciso crear la declaración de estilo. Algunos componentes, como TextArea y TextInput, tienen de forma predeterminada declaraciones de estilo de clase predefinidas, ya que sus propiedades `borderStyle` y `backgroundColor` deben personalizarse.

ATENCIÓN

Si cambia una hoja de estilos de clase, asegúrese de añadir los estilos que desea de la hoja de estilos anterior; de no hacerlo, se sobrescribirán.

En el siguiente código se comprueba primero si el tema actual ya tiene una declaración de estilo definida para el componente CheckBox; de no ser así, se crea una nueva. A continuación, el código utiliza el método `setStyle()` para definir una propiedad de estilo para la declaración de estilo de CheckBox (en este caso, "color" establece el color azul en todo el texto de la etiqueta de la casilla de verificación):

```
if (_global.styles.CheckBox == undefined) {
    _global.styles.CheckBox = new mx.styles.CSSStyleDeclaration();
}
_global.styles.CheckBox.setStyle("color", 0x0000FF);
```

Para ver una tabla de las propiedades de estilo que pueden definirse en el componente CheckBox, consulte “Utilización de estilos con el componente CheckBox” en *Referencia del lenguaje de componentes*.

Las configuraciones de estilo personalizado tienen prioridad sobre las configuraciones de estilo heredado o global. Para ver una lista de la prioridad de estilos, consulte “Utilización de un estilo global y estilos personalizados y de clase en un mismo documento” en la página 97.

Definición de estilos heredados en un contenedor

Un *estilo heredado* es un estilo que hereda su valor de los componentes principales en la jerarquía de MovieClip del documento. Si un estilo o color de texto no se ha establecido en un nivel de instancia, personalizado o de clase, Flash busca el valor del estilo en la jerarquía de MovieClip. Así, si se establecen estilos en un componente contenedor, los componentes contenidos heredarán esta configuración de estilo.

Los estilos siguientes son estilos heredados:

- fontFamily
- fontSize
- fontStyle
- fontWeight
- textAlign
- textIndent
- Todos los estilos de color de un solo valor (por ejemplo, themeColor es un estilo heredado mientras que alternatingRowColors no lo es)

Style Manager indica a Flash si un estilo hereda su valor. También se puede añadir estilos adicionales en tiempo de ejecución como estilos heredados. Para más información, consulte [Clase StyleManager](#) en *Referencia del lenguaje de componentes*.

NOTA

Una de las principales diferencias entre la implementación de estilos en los componentes de Flash y las hojas de estilos en cascada en las páginas HTML es que en los componentes de Flash no se admite el valor de herencia (`inherit`) de la hoja de estilos en cascada. Los estilos se heredan o no a través del diseño del componente.

Los estilos heredados tienen prioridad sobre los estilos globales. Para ver una lista de la prioridad de estilos, consulte “Utilización de un estilo global y estilos personalizados y de clase en un mismo documento” en la página 97.

En el siguiente ejemplo se muestra cómo utilizar los estilos heredados con un componente Accordion, disponible en Flash Professional 8. (Tanto Flash Basic 8 como Flash Professional 8 ofrecen la característica de estilos heredados.)

Para crear un componente Accordion con estilos que heredarán los componentes de los paneles Accordion individuales:

1. Abra un archivo FLA nuevo.
2. Arrastre un componente Accordion desde el panel Componentes al escenario.
3. Use el inspector de propiedades para asignar el nombre y el tamaño al componente Accordion. Para este ejemplo, asigne al componente el nombre de instancia **accordion**.
4. Arrastre un componente TextInput y un componente Button desde el panel Componentes a la biblioteca.

Los componentes que se arrastran a la biblioteca estarán disponibles en el script en tiempo de ejecución.

5. Añada el siguiente código ActionScript al primer fotograma de la línea de tiempo:

```
var section1 = accordion.createChild(mx.core.View, "section1", {label:
    "First Section"});
var section2 = accordion.createChild(mx.core.View, "section2", {label:
    "Second Section"});

var input1 = section1.createChild(mx.controls.TextInput, "input1");
var button1 = section1.createChild(mx.controls.Button, "button1");

input1.text = "Text Input";
button1.label = "Button";
button1.move(0, input1.height + 10);

var input2 = section2.createChild(mx.controls.TextInput, "input2");
var button2 = section2.createChild(mx.controls.Button, "button2");

input2.text = "Text Input";
button2.label = "Button";
button2.move(0, input2.height + 10);
```

El código anterior añade dos elementos secundarios al componente Accordion y carga en cada elemento secundario un control TextInput y un control Button, que se usan en este ejemplo para ilustrar la herencia de estilos.

6. Seleccione Control > Probar película para ver el documento antes de añadir herencia de estilos.
7. Añada el siguiente código ActionScript al final del script del primer fotograma:

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

8. Seleccione Control > Probar película para ver los cambios.

Observe que la configuración de `fontStyle` no sólo afecta al componente Accordion, sino también al texto asociado con los componentes TextInput y Button dentro del componente Accordion.

Utilización de un estilo global y estilos personalizados y de clase en un mismo documento

Si define un estilo sólo en un sitio del documento, Flash utilizará dicha definición cuando necesite saber el valor de una propiedad. No obstante, un documento de Flash puede tener diversas configuraciones de estilo (propiedades de estilo establecidas directamente en instancias de componente, declaraciones de estilo personalizado, declaraciones de estilo de clase predeterminado, estilos heredados y una declaración de estilo global). En esta situación, Flash determina el valor de una propiedad buscando su definición en todos los lugares mencionados, siguiendo un orden específico.

Flash busca estilos en el orden siguiente hasta que encuentra un valor:

1. Busca una propiedad de estilo en la instancia de componente.
2. Busca en la propiedad `styleName` de la instancia para ver si tiene una declaración de estilo personalizado asignada.
3. Busca la propiedad en una declaración de estilo de clase predeterminado.
4. Si el estilo es uno de los estilos heredados, Flash busca en la jerarquía de elementos principales un valor heredado.
5. Busca el estilo en la declaración de estilo global.
6. Si la propiedad aún no está definida, tiene el valor `undefined`.

Propiedades de estilo de color

Las propiedades de estilo relacionadas con el color se comportan de forma diferente a las demás propiedades de estilo. Todas estas propiedades tienen un nombre que termina en “Color”; por ejemplo, `backgroundColor`, `disabledColor` y `color`. Cuando se cambian las propiedades de estilo relativas al color, el color de la instancia y de todas las instancias secundarias correspondientes cambiará inmediatamente. Los cambios restantes en la propiedad de estilo sólo indican que el objeto se debe volver a dibujar y que los cambios se producirán a partir del fotograma siguiente.

El valor de una propiedad de estilo de color puede ser un número, una cadena o un objeto. Si se trata de un número, representa el valor RVA del color como número hexadecimal (0xRRGGBB). Si el valor es una cadena, debe ser un nombre de color.

Los nombres de color son cadenas que se asignan a los colores que se utilizan con mayor frecuencia. Puede añadir nuevos nombres de colores a través de Style Manager (consulte [Clase StyleManager](#) en *Referencia del lenguaje de componentes*). En la tabla siguiente se enumeran los nombres de color predeterminados:

Nombre del color	Valor
black	0x000000
white	0xFFFFFFFF
red	0xFF0000
green	0x00FF00
blue	0x0000FF
magenta	0xFF00FF
yellow	0xFFFF00
cyan	0x00FFFF
haloGreen	0x80FF4D
haloBlue	0x2BF5F5
haloOrange	0xFFC200

NOTA

Si el nombre del color no está definido, es posible que el componente no se dibuje correctamente.

Puede utilizar cualquier identificador de ActionScript válido para crear sus propios nombres de color (por ejemplo, "WindowText" o "ButtonText"). Use Style Manager para definir nuevos colores, como se indica a continuación:

```
mx.styles.StyleManager.registerColorName("special_blue", 0x0066ff);
```

La mayoría de los componentes no pueden gestionar un objeto como valor de la propiedad de estilo de color. No obstante, determinados componentes pueden gestionar objetos de color que representan degradados u otras combinaciones de color. Para más información, consulte la sección "Utilización de estilos" de cada entrada de componente en *Referencia del lenguaje de componentes*.

Puede utilizar las declaraciones de estilo de clase y los nombres de color para controlar fácilmente los colores del texto y los símbolos que se muestran en la pantalla. Por ejemplo, si desea incluir una pantalla de configuración de la visualización que tenga la apariencia de Microsoft Windows, defina nombres de color como `ButtonText` y `WindowText`, y declaraciones de estilo de clase como `Button`, `CheckBox` y `Window`.

NOTA

Algunos componentes proporcionan propiedades de estilo que son una matriz de colores, como `alternatingRowColors`. Debe establecer estos estilos únicamente como una matriz de valores numéricos de RVA, no como nombres de colores.

Personalización de animaciones de componentes

Varios componentes, como los componentes `Accordion`, `ComboBox` y `Tree`, proporcionan animación para ilustrar la transición entre estados del componente (por ejemplo, al cambiar entre elementos secundarios de `Accordion`, desplegar la lista desplegable del componente `ComboBox`, y expandir o contraer las carpetas del componente `Tree`). Además, los componentes proporcionan animación relacionada con la selección y anulación de la selección de un elemento, como las filas de una lista.

Puede controlar los detalles de estas animaciones mediante los siguientes estilos:

Estilo de animación	Descripción
<code>openDuration</code>	La duración de la transición de suavizado de apertura de los componentes <code>Accordion</code> , <code>ComboBox</code> y <code>Tree</code> , en milisegundos. El valor predeterminado es 250.
<code>openEasing</code>	Una referencia a una función de interpolación que controla la animación del estado en los componentes <code>Accordion</code> , <code>ComboBox</code> y <code>Tree</code> . La ecuación predeterminada usa una función sinusoidal entrante/saliente.
<code>popupDuration</code>	La duración de la transición cuando se abre un menú en el componente <code>Menu</code> , en milisegundos. El valor predeterminado es 150. Sin embargo, debe tenerse en cuenta que la animación siempre usa la función sinusoidal entrante/saliente predeterminada.

Estilo de animación	Descripción
<code>selectionDuration</code>	La duración de la transición en componentes <code>ComboBox</code> , <code>DataGrid</code> , <code>List</code> y <code>Tree</code> de un estado normal a un estado seleccionado o de un estado seleccionado al estado normal, en milisegundos. El valor predeterminado es 200.
<code>selectionEasing</code>	Una referencia a una función de interpolación que controla la animación de la selección en componentes <code>ComboBox</code> , <code>DataGrid</code> , <code>List</code> y <code>Tree</code> . Este estilo sólo se aplica a la transición de un estado normal a un estado seleccionado. La ecuación predeterminada usa una función sinusoidal entrante/saliente.

El paquete `mx.transitions.easing` proporciona seis clases para controlar el suavizado:

Clase Easing	Descripción
<code>Back</code>	Se extiende más allá del rango de la transición en uno de los extremos (o ambos) una vez para proporcionar un ligero efecto de desbordamiento.
<code>Bounce</code>	Proporciona un efecto de rebote completamente dentro del rango de la transición en uno o ambos extremos. El número de rebotes está relacionado con la duración: a mayor duración, más rebotes.
<code>Elastic</code>	Proporciona un efecto elástico que está fuera del rango de la transición en uno o ambos extremos. La duración no afecta al grado de elasticidad.
<code>None</code>	Proporciona un movimiento uniforme de principio a fin sin efectos, ralentización ni aceleración. Esta transición también se suele denominar <i>transición lineal</i> .
<code>Regular</code>	Produce un movimiento más lento en uno de los extremos (o en ambos) para proporcionar un efecto de aceleración, de ralentización o ambos.
<code>Strong</code>	Produce un movimiento mucho más lento en uno de los extremos (o en ambos). Este efecto es similar a <code>Regular</code> pero mucho más pronunciado.

Cada una de las clases del paquete `mx.transitions.easing` proporciona los tres métodos de suavizado siguientes:

Método de suavizado	Descripción
<code>easeIn</code>	Proporciona el efecto de suavizado al principio de la transición.
<code>easeOut</code>	Proporciona el efecto de suavizado al final de la transición.
<code>easeInOut</code>	Proporciona el efecto de suavizado al principio y al final de la transición.

Como los métodos de suavizado son métodos estáticos de las clases de suavizado, no tiene que crear instancias de dichas clases. Los métodos se usan en llamadas a `setStyle()`, como en el siguiente ejemplo.

```
import mx.transitions.easing.*;
trace("_global.styles.Accordion = " + _global.styles.Accordion);
_global.styles.Accordion.setStyle("openDuration", 1500);
_global.styles.Accordion.setStyle("openEasing", Bounce.easeOut);
```

NOTA

La ecuación predeterminada usada por todas las transiciones no está disponible en las clases de suavizado enumeradas arriba. Para especificar que un componente debe usar el método de suavizado predeterminado después de que se haya especificado otro método de suavizado, llame a `setStyle("openEasing", null)`.

Para más información, consulte [“Aplicación de métodos de suavizado a componentes”](#) en *Referencia del lenguaje de componentes*.

Obtención de los valores de las propiedades de estilo

Para recuperar el valor de una propiedad de estilo, use `UIObject.getStyle()`. Cada componente que sea una subclase de `UIObject` (lo que incluye todos los componentes de la versión 2, salvo los componentes multimedia) hereda el método `getStyle()`. Esto significa que puede llamar a `getStyle()` desde cualquier instancia de componente, igual que puede llamar a `setStyle()` desde cualquier instancia de componente.

El código siguiente obtiene el valor del estilo de `themeColor` y se lo asigna a la variable `oldStyle`:

```
var myCheckBox:mx.controls.CheckBox;
var oldFontSize:Number

oldFontSize = myCheckBox.getStyle("fontSize");
trace(oldFontSize);
```

Aplicación de aspectos a los componentes

Los aspectos son símbolos de clip de película que un componente utiliza para mostrar su apariencia. La mayoría de los aspectos contienen formas que representan la apariencia del componente. Algunos aspectos contienen sólo código ActionScript que dibuja el componente en el documento.

Los componentes de la versión 2 son clips compilados, cuyos elementos no se pueden ver en la biblioteca. No obstante, la instalación de Flash incluye archivos FLA que contienen todos los aspectos de los componentes. Estos archivos FLA reciben el nombre de *temas*. Cada tema tiene un comportamiento y una apariencia diferentes, aunque contiene aspectos con los mismos nombres de símbolo e identificadores de vínculo. Esto le permite arrastrar un tema al escenario de un documento para cambiar su apariencia. Los archivos FLA del tema también pueden utilizarse para editar aspectos de componente. Los aspectos se encuentran en la carpeta Themes del panel Biblioteca de cada archivo FLA del tema. Para más información sobre los temas, consulte [“Temas” en la página 115](#).

Cada componente está formado por varios aspectos. Por ejemplo, la flecha abajo del subcomponente ScrollBar consta de cuatro aspectos: ScrollDownArrowDisabled, ScrollDownArrowDown, ScrollDownArrowOver y ScrollDownArrowUp. El componente ScrollBar completo usa 13 símbolos de aspecto distintos.

Algunos componentes comparten aspectos; por ejemplo, los componentes que usan barras de desplazamiento (como ComboBox, List y ScrollPane) comparten los aspectos de la carpeta ScrollBar Skins. Puede editar los aspectos ya existentes y crear aspectos nuevos para cambiar la apariencia de los componentes.

El archivo AS que define cada clase de componente contiene código que carga aspectos específicos del componente. Cada aspecto de componente corresponde a una propiedad de aspecto asignada al identificador de vinculación de un símbolo de aspecto. Por ejemplo, el estado presionado (abajo) de la flecha abajo del componente ScrollBar tiene el nombre de propiedad de aspecto `downArrowDownName`. El valor predeterminado de la propiedad `downArrowDownName` es `"ScrollDownArrowDown"`, que es el identificador de vinculación del símbolo de aspecto del archivo FLA del tema. Puede editar aspectos existentes y aplicarlos a todos los componentes que usen el aspecto editando el símbolo del aspecto y dejando el identificador de vinculación existente. Puede crear nuevos aspectos y aplicarlos a instancias de componente específicas estableciendo las propiedades de aspecto para una instancia de componente. No tiene que editar el archivo AS del componente para cambiar sus propiedades de aspecto; puede pasar simplemente los valores de propiedad del aspecto a la función constructora del componente cuando éste se crea en el documento.

Las propiedades de aspecto de un componente se enumeran en la entrada del componente en el Diccionario de componentes. Por ejemplo, las propiedades de aspecto para el componente Button se encuentran en: Referencia del lenguaje de componentes > Componente Button > Personalización del componente Button > Utilización de aspectos con el componente Button.

En función de lo que desee hacer, puede elegir una de las maneras siguientes para aplicar un aspecto a un componente. Se enumeran los distintos enfoques desde el más sencillo al más difícil.

- Para cambiar los aspectos asociados a todas las instancias de un componente específico en un documento individual, copie y modifique elementos de aspecto individuales. (Véase [“Edición de aspectos de componente en un documento” en la página 104.](#))
Este método de aplicación de aspectos está recomendado para principiantes, ya que no requiere crear ningún script.
- Para sustituir todos los aspectos de un documento por un conjunto nuevo (en el que cada tipo de componente comparta la misma apariencia), aplique un tema. (Véase [“Temas” en la página 115.](#))
Este método de aplicación de aspectos es recomendable para aplicar una apariencia uniforme a todos los componentes y a varios documentos.
- Para vincular el color de un elemento de aspecto a una propiedad de estilo, añada código ActionScript al aspecto para registrarlo como un elemento de aspecto de color. (Véase [“Vinculación del color de un aspecto a los estilos” en la página 106.](#))
- Para utilizar aspectos diferentes para varias instancias del mismo componente, cree nuevos aspectos y establezca las propiedades de aspecto. (Véase [“Creación de nuevos aspectos de componente” en la página 105](#) y [“Aplicación de nuevos aspectos a un componente” en la página 108.](#))
- Para cambiar aspectos de un subcomponente (como una barra de desplazamiento de un componente List), ponga el componente en una subclase. (Véase [“Aplicación de aspectos nuevos a un subcomponente” en la página 109.](#))
- Para cambiar aspectos de un subcomponente a los que no se tiene acceso directo desde el componente principal (como un componente List de un componente ComboBox), sustituya las propiedades del aspecto en el prototipo. (Véase [“Cambio de propiedades de aspecto en un subcomponente” en la página 113.](#))

Edición de aspectos de componente en un documento

Para editar los aspectos asociados con todas las instancias de un componente específico en un documento individual, copie los símbolos de aspecto del tema al documento y realice los cambios que desee en los gráficos.

El procedimiento descrito a continuación es muy similar a crear y aplicar un tema nuevo (véase “Temas” en la [página 115](#)). La diferencia principal es que este procedimiento describe la copia de símbolos directamente desde el tema que ya se está utilizando a un documento individual y la edición de sólo un número reducido de todos los aspectos disponibles. Esto es apropiado cuando todos los cambios están en un documento individual y al modificar aspectos para sólo unos pocos componentes. Si los aspectos editados se van a compartir en varios documentos o abarcan cambios en varios componentes, puede que sea más fácil editar los aspectos si crea un nuevo tema.

Encontrará un artículo sobre la aplicación de aspectos avanzada en el Centro de desarrolladores de Macromedia www.macromedia.com/devnet/mx/flash/articles/skinning_2004.html (en inglés).

Para editar aspectos de componente en un documento:

1. Si ya aplicó el tema Sample a un documento, vaya al paso 5.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo SampleTheme 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 la [página 115](#).
3. En el panel Biblioteca del tema, seleccione Flash UI Components 2/Themes/MMDefault y arrastre la carpeta Assets de cualquier componente del documento a la biblioteca para el documento.

Por ejemplo, arrastre la carpeta RadioButton Assets a la biblioteca ThemeApply fla.
4. Si arrastró carpetas Assets individuales a la biblioteca, asegúrese de que el símbolo Assets de cada componente está establecido en Exportar en primer fotograma.

Por ejemplo, la carpeta Assets para el componente RadioButton se denomina RadioButton Assets; tiene un símbolo denominado RadioButtonAssets, que contiene todos los símbolos de elementos individuales. Si establece Exportar en primer fotograma en el símbolo de RadioButtonAssets, todos los símbolos de activos individuales también se exportarán en el primer fotograma.

5. Haga doble clic en el símbolo de aspecto que desee modificar para abrirlo en el modo de edición de símbolos.

Por ejemplo, abra el símbolo States/RadioFalseDisabled.

6. Modifique el símbolo o elimine los gráficos y cree gráficos nuevos.

Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).

Por ejemplo, cambie el color del círculo interior a gris claro.

7. Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información situada en la parte superior del escenario para volver al modo de edición de documentos.
8. Repita los pasos 5 a 7 hasta que haya terminado de editar todos los aspectos que desee cambiar.

NOTA

La previsualización dinámica de los componentes del escenario no reflejará los aspectos editados.

9. Seleccione Control > Probar película.

En este ejemplo, asegúrese de que tiene una instancia de RadioButton en el escenario y establezca su propiedad `enabled` en `false` en el panel Acciones para ver la nueva apariencia de RadioButton desactivado.

Creación de nuevos aspectos de componente

Si desea utilizar un aspecto determinado para una instancia de un componente, pero dispone de otro aspecto de otra instancia del componente, debe abrir un archivo FLA del tema y crear un símbolo de aspecto nuevo. Los componentes se han diseñado para facilitar el uso de aspectos diferentes para instancias diferentes.

Para crear un aspecto nuevo:

1. Seleccione Archivo > Abrir y abra el archivo FLA del tema que desea utilizar como plantilla.
2. Seleccione Archivo > Guardar como y, a continuación, seleccione un nombre exclusivo como `MyTheme fla`.
3. Seleccione los aspectos que desea editar (en este ejemplo, `RadioTrueUp`).

Los aspectos se encuentran en la carpeta `Themes/MMDefault/Componente Assets` (en este ejemplo, `Themes/MMDefault/RadioButton Assets/States`).

4. Seleccione Duplicar en el menú Opciones de Biblioteca (o haga clic con el botón derecho del ratón en el símbolo) y asigne un nombre único al símbolo, como **MyRadioTrueUp**.
5. Haga clic en Avanzado en el cuadro diálogo Propiedades de símbolo y seleccione Exportar para ActionScript.
Se introducirá automáticamente un identificador de vinculación que coincida con el nombre de símbolo.
6. Haga doble clic en el nuevo aspecto de la biblioteca para abrirlo en modo de edición de símbolos.
7. Modifique el clip de película o elimínelo y cree uno nuevo.
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
8. Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información situada en la parte superior del escenario para volver al modo de edición de documentos.
9. Seleccione Archivo > Guardar, pero no cierre MyTheme.flc. Ahora debe crear un documento nuevo donde aplicar el aspecto editado a un componente.
Para más información, consulte [“Aplicación de nuevos aspectos a un componente” en la página 108](#), [“Aplicación de aspectos nuevos a un subcomponente” en la página 109](#) o [“Cambio de propiedades de aspecto en un subcomponente” en la página 113](#).

NOTA

Flash no muestra los cambios realizados a los aspectos de componente al mostrar componentes en el escenario mediante la previsualización dinámica.

Vinculación del color de un aspecto a los estilos

La versión 2 de la arquitectura de componentes facilita en gran medida la vinculación de un elemento visual a un estilo establecido en el componente mediante el aspecto. Para registrar una instancia de un clip de película para un estilo, o un elemento de aspecto completo para un estilo, añada código ActionScript en la línea de tiempo del aspecto para llamar a `mx.skins.ColoredSkinElement.setColorStyle(targetMovieClip, styleName)`.

Para vincular un aspecto a una propiedad de estilo:

1. Si ya aplicó el tema Sample a un documento, vaya al paso 5.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo SampleTheme.flc.

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 la página 115](#).

3. En el panel Biblioteca del tema, seleccione Flash UI Components 2/Themes/MMDefault y arrastre la carpeta Assets de cualquier componente del documento a la biblioteca para el documento.

Por ejemplo, arrastre la carpeta RadioButton Assets a la biblioteca de destino.

4. Si arrastró carpetas de elementos individuales a la biblioteca, asegúrese de que el símbolo Assets de cada componente está establecido en Exportar en primer fotograma.

Por ejemplo, la carpeta Assets para el componente RadioButton se denomina RadioButton Assets; tiene un símbolo denominado RadioButtonAssets, que contiene todos los símbolos de elementos individuales. Si establece Exportar en primer fotograma en el símbolo de RadioButtonAssets, todos los símbolos de activos individuales también se exportarán en el primer fotograma.

5. Haga doble clic en el símbolo de aspecto que desee modificar para abrirlo en el modo de edición de símbolos.

Por ejemplo, abra el símbolo States/RadioFalseDisabled.

6. Si el elemento que se va a colorear es un símbolo de gráfico y no una instancia de clip de película, use Modificar > Convertir en símbolo para convertirlo en una instancia de clip de película.

Para este ejemplo, cambie el gráfico central, que es una instancia del símbolo gráfico RadioShape1, por un símbolo de clip de película; a continuación, asígnele el nombre **Inner Circle**. No tiene que seleccionar Exportar para ActionScript.

Es recomendable, pero no necesario, mover el símbolo de clip de película recién creado a la carpeta Elements de los elementos de componente que se están editando.

7. Si convirtió un símbolo gráfico en una instancia de clip de película en el paso anterior, asigne a esa instancia un nombre de forma que se pueda usar en código ActionScript.

Para este ejemplo, asigne a la instancia el nombre **innerCircle**.

8. Añada código ActionScript para registrar el elemento de aspecto o una instancia de clip de película que contenga como un elemento de aspecto coloreado.

Por ejemplo, añada el código siguiente a la línea de tiempo del elemento de aspecto.

```
mx.skins.ColoredSkinElement.setColorStyle(innerCircle,  
"symbolBackgroundDisabledColor");
```

En este ejemplo se usa un color que ya corresponde a un nombre de estilo existente en el estilo de Sample. Siempre que sea posible, es mejor usar los nombres de estilo correspondientes a las normas oficiales de hojas de estilos en cascada (CSS) o estilos proporcionados por los temas Halo y Sample.

9. Repita los pasos 5 a 8 hasta que haya terminado de editar todos los aspectos que desee cambiar.

Para este ejemplo, repita estos pasos para el aspecto `RadioTrueDisabled`, pero en lugar de convertir el gráfico existente en un clip de película, elimine el gráfico y arrastre el símbolo de Inner Circle existente al elemento de aspecto `RadioTrueDisabled`.

10. Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información situada en la parte superior del escenario para volver al modo de edición de documentos.

11. Arrastre una instancia del componente al escenario.

Para este ejemplo, arrastre dos componentes `RadioButton` al escenario, establezca uno en estado seleccionado y use código `ActionScript` para establecer ambos en estado desactivado para ver los cambios.

12. Añada código `ActionScript` al documento para establecer la nueva propiedad de estilo en las instancias de componente o en el nivel global.

Para este ejemplo, establezca la propiedad en el nivel global de la manera siguiente:

```
_global.style.setStyle("symbolBackgroundDisabledColor", 0xD9D9D9);
```

13. Seleccione `Control > Probar película`.

Aplicación de nuevos aspectos a un componente

Cuando haya creado un aspecto nuevo, tendrá que aplicarlo a un componente de un documento. Puede utilizar el método `createClassObject()` para crear dinámicamente las instancias del componente, o bien puede poner manualmente las instancias del componente en el escenario. Existen dos maneras diferentes de aplicar aspectos a instancias de un componente, según cómo añada los componentes al documento.

Para crear dinámicamente un componente y aplicar un aspecto nuevo:

1. Seleccione `Archivo > Nuevo` para crear un documento de Flash nuevo.
2. Seleccione `Archivo > Guardar` y asígnele un nombre exclusivo como `DynamicSkinning fla`.
3. Arrastre los componentes desde el panel Componentes a la librería, incluido el componente cuyo aspecto ha editado (`RadioButton` en este ejemplo).

De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en el documento.

4. Arrastre MyRadioTrueUp y cualquier otro símbolo que haya personalizado desde MyTheme fla a la biblioteca de DynamicSkinning fla.

De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en el documento.

5. Abra el panel Acciones e introduzca lo siguiente en el fotograma 1:

```
import mx.controls.RadioButton;  
createClassObject(RadioButton, "myRadio", 0,  
    {trueUpIcon:"MyRadioTrueUp", label: "My Radio Button"});
```

6. Seleccione Control > Probar película.

Para añadir manualmente un componente al escenario y aplicar un aspecto nuevo:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **ManualSkinning fla**.
3. Arrastre los componentes desde el panel Componentes hasta el escenario, incluido el componente cuyo aspecto ha editado (en este ejemplo, se trata de RadioButton).
4. Arrastre MyRadioTrueUp y cualquier otro símbolo que haya personalizado desde MyTheme fla a la biblioteca de ManualSkinning fla.

De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en el documento.

5. Seleccione el componente RadioButton en el escenario y abra el panel Acciones.

6. Asocie el código siguiente a la instancia RadioButton:

```
onClipEvent(initialize){  
    trueUpIcon = "MyRadioTrueUp";  
}
```

7. Seleccione Control > Probar película.

Aplicación de aspectos nuevos a un subcomponente

En determinadas situaciones, es posible que desee modificar los aspectos de un subcomponente de un componente y se encuentre con que las propiedades de dichos aspectos no están disponibles directamente (por ejemplo, no hay forma directa de modificar los aspectos de la barra de desplazamiento de un componente List). El código siguiente permite acceder a los aspectos de barra de desplazamiento. Todas las barras de desplazamiento que se creen después de ejecutar este código también tendrán los aspectos nuevos.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en *Referencia del lenguaje de componentes*.

Para aplicar un aspecto nuevo a un subcomponente:

1. Siga los pasos indicados en [“Creación de nuevos aspectos de componente” en la página 105](#), pero edite un aspecto de la barra de desplazamiento.
En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nuevo nombre **MyScrollDownArrowDown**.
2. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
3. Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **SubcomponentProject.fla**.
4. Arrastre un componente List desde el panel Componentes a la biblioteca.
De esta manera, se añade el componente al panel Biblioteca, aunque no se vuelve visible en el documento.
5. Arrastre MyScrollDownArrowDown y cualquier otro símbolo que haya editado desde MyTheme.fla a la biblioteca de SubcomponentProject.fla.
De esta manera, se añade el símbolo al panel Biblioteca, aunque no se vuelve visible en el documento.
6. Siga uno de estos procedimientos:

- Si desea cambiar todas las barras de desplazamiento de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List;
import mx.controls.scrollClasses.ScrollBar;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
```

A continuación puede introducir el código siguiente en el fotograma 1 para crear una lista dinámicamente:

```
createClassObject(List, "myListBox", 0, {dataProvider:
    ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

O bien, puede arrastrar un componente List desde la biblioteca al escenario.

- Si desea cambiar una barra de desplazamiento específica de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
var oldName = ScrollBar.prototype.downArrowDownName;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
createClassObject(List, "myList1", 0, {dataProvider: ["AL", "AR", "AZ",
    "CA", "HI", "ID", "KA", "LA", "MA"]});
myList1.redraw(true);
ScrollBar.prototype.downArrowDownName = oldName;
```

NOTA

Establezca datos suficientes para que aparezcan las barras de desplazamiento o establezca la propiedad `vScrollPolicy` en `true`.

7. Seleccione Control > Probar película.

También puede definir los aspectos de un subcomponente para todos los componentes de un documento definiendo la propiedad del aspecto en el objeto `prototipo` del subcomponente en la sección `#initclip` del símbolo del aspecto.

Para utilizar `#initclip` a fin de aplicar un aspecto editado a todos los componentes de un documento:

1. Siga los pasos indicados en [“Creación de nuevos aspectos de componente” en la página 105](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto `ScrollDownArrowDown` y asígnele el nombre nuevo `MyScrollDownArrowDown`.
2. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo. Guárdelo con un nombre exclusivo, como `SkinsInitExample fla`.
3. Seleccione el símbolo `MyScrollDownArrowDown` en la biblioteca del ejemplo de la biblioteca del tema editado y arrástrelo a la biblioteca de `SkinsInitExample fla`. Esto añade el símbolo a la biblioteca sin hacerlo visible en el escenario.
4. Seleccione `MyScrollDownArrowDown` en la biblioteca `SkinsInitExample fla` y seleccione Vinculación en el menú Opciones de Biblioteca.
5. Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar. Exportar en primer fotograma debería activarse automáticamente; de no ser así, active esta opción.
6. Haga doble clic en `MyScrollDownArrowDown` de la biblioteca para abrirlo en el modo de edición de símbolos.
7. Introduzca el código siguiente en el fotograma 1 del símbolo `MyScrollDownArrowDown`:

```
#initclip 10
    import mx.controls.scrollClasses.ScrollBar;
    ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
#endinitclip
```

8. Para añadir un componente List al documento, siga uno de estos procedimientos:
- Arrastre un componente List desde el panel Componentes al escenario. Introduzca los parámetros de etiqueta que haga falta para que aparezca la barra de desplazamiento vertical.
 - Arrastre un componente List desde el panel Componentes a la biblioteca. Introduzca el código siguiente en el fotograma 1 de la línea de tiempo principal de SkinsInitExample.fla:

```
createClassObject(mx.controls.List, "myListBox1", 0, {dataProvider:  
    ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

NOTA

Añada información suficiente para que aparezca la barra de desplazamiento vertical, o defina `vScrollPolicy` en `true`.

En el siguiente ejemplo se explica cómo aplicar aspectos a algo que ya está en el escenario. En este ejemplo sólo se aplica un aspecto a barras de desplazamiento de componentes List; no se aplican aspectos a las barras de desplazamiento de componentes TextArea o ScrollPane.

Para utilizar #initclip a fin de aplicar un aspecto editado a componentes específicos de un documento:

1. Siga los pasos indicados en [“Edición de aspectos de componente en un documento” en la página 104](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.
2. Seleccione Archivo > Nuevo y cree un documento de Flash.
3. Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyVScrollTest.fla**.
4. Arrastre MyScrollDownArrowDown desde la biblioteca del tema a la biblioteca MyVScrollTest.fla.
5. Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyVScrollBar**.
6. Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar. Exportar en primer fotograma debería activarse automáticamente; de no ser así, active esta opción.

7. Introduzca el código siguiente en el fotograma 1 del símbolo MyVScrollBar:

```
#initclip 10  
import MyVScrollBar  
Object.registerClass("VScrollBar", MyVScrollBar);  
#endinitclip
```

8. Arrastre un componente List desde el panel Componentes al escenario.

9. En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.
10. Seleccione Archivo > Guardar.
11. Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo.
12. Introduzca el código siguiente:

```
import mx.controls.VScrollBar
import mx.controls.List
class MyVScrollBar extends VScrollBar{
    function init():Void{
        if (_parent instanceof List){
            downArrowDownName = "MyScrollDownArrowDown";
        }
        super.init();
    }
}
```
13. Seleccione Archivo > Guardar y guarde este archivo como **MyVScrollBar.as**.
14. Haga clic en un área vacía del escenario y, en el inspector de propiedades, haga clic en el botón Configuración de publicación.
15. Haga clic en el botón Configuración de la versión ActionScript.
16. Haga clic en el botón Añadir nueva ruta (+) para añadir una nueva ruta de clases y seleccione el botón Destino para desplazarse hasta la ubicación del archivo MyVScrollBar.as en el disco duro.
17. Seleccione Control > Probar película.

Cambio de propiedades de aspecto en un subcomponente

Si un componente no admite directamente variables del aspecto, puede crear una subclase del componente y sustituir sus aspectos. Por ejemplo, el componente ComboBox no admite directamente la aplicación de aspectos a su lista desplegable, ya que utiliza un componente List como lista desplegable.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en *Referencia del lenguaje de componentes*.

Para aplicar un aspecto a un subcomponente:

1. Siga los pasos indicados en [“Edición de aspectos de componente en un documento” en la página 104](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.
2. Seleccione Archivo > Nuevo y cree un documento de Flash.
3. Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyComboTest.fla**.
4. Arrastre MyScrollDownArrowDown desde la biblioteca del tema a la biblioteca de MyComboTest.fla.
Esto añade el símbolo a la biblioteca, pero no lo hace visible en el escenario.
5. Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyComboBox**.
6. Seleccione la casilla de verificación Exportar para ActionScript y haga clic en Aceptar.
Exportar en primer fotograma debería activarse automáticamente; de no ser así, active esta opción.
7. Introduzca el código siguiente en el panel Acciones, en el fotograma 1 del símbolo de MyComboBox:

```
#initclip 10
    import MyComboBox
    Object.registerClass("ComboBox", MyComboBox);
#endinitclip
```
8. Cuando haya terminado de editar el símbolo, haga clic en el botón Atrás situado en la parte izquierda de la barra de información en el área superior del escenario para volver al modo de edición de documentos.
9. Arrastre un componente ComboBox al escenario.
10. En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.
11. Seleccione Archivo > Guardar.
12. Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo.

13. Introduzca el código siguiente:

```
import mx.controls.ComboBox
import mx.controls.scrollClasses.ScrollBar
class MyComboBox extends ComboBox{
    function getDropdown():Object{
        var oldName = ScrollBar.prototype.downArrowDownName;
        ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
        var r = super.getDropdown();
        ScrollBar.prototype.downArrowDownName = oldName;
        return r;
    }
}
```

14. Seleccione Archivo > Guardar y guarde este archivo como **MyComboBox.as**.

15. Vuelva al archivo MyComboTest fla.

16. Haga clic en un área vacía del escenario y, en el inspector de propiedades, haga clic en el botón Configuración de publicación.

17. Haga clic en el botón Configuración de la versión ActionScript.

18. Haga clic en el botón Añadir nueva ruta (+) para añadir una nueva ruta de clases y seleccione el botón Destino para desplazarse hasta la ubicación del archivo MyComboBox.as en el disco duro.

19. Seleccione Control > Probar película.

Temas

Los temas son conjuntos de estilos y de aspectos. El tema predeterminado de Flash se llama Halo (HaloTheme fla). El tema Halo permite ofrecer una experiencia interactiva y expresiva a los usuarios. Flash incluye temas adicionales como, por ejemplo, Sample (SampleTheme fla). El tema Sample ofrece un ejemplo de cómo se pueden utilizar más estilos para la personalización. (El tema Halo no usa todos los estilos incluidos en el tema Sample.) Los archivos del tema se encuentran en las siguientes carpetas de una instalación predeterminada:

- En Windows: C:\Archivos de programa\Macromedia\Flash 8\idioma\Configuration\ComponentFLA\
- En Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/

Puede crear temas nuevos y aplicarlos a una aplicación para cambiar la apariencia de todos los componentes. Por ejemplo, puede crear temas que imiten la apariencia nativa del sistema operativo.

Los componentes usan aspectos (símbolos de gráficos o clips de película) para mostrar sus apariencias. El archivo AS que define cada componente contiene código que carga aspectos específicos del componente. Puede crear fácilmente un tema nuevo haciendo una copia del tema Halo o el tema Sample y modificando los gráficos de los aspectos.

Un tema también puede contener un nuevo conjunto de valores predeterminados del estilo. Para crear una declaración de estilo global o cualquier otra declaración de estilo adicional, es preciso escribir código ActionScript. Para más información, consulte [“Modificación de los valores predeterminados de las propiedades de estilo de un tema”](#) en la página 120.

Cambio entre temas

Macromedia Flash instala dos temas: Halo y Sample. Observará que la información de referencia de cada componente contiene una tabla de propiedades de estilo que pueden establecerse en cada tema (o en ambos). Así pues, cuando lea una tabla de propiedades de estilo, como la correspondiente al componente Button que encontrará en “Utilización de estilos con el componente Button” en *Referencia del lenguaje de componentes*, observe qué tema es compatible con el estilo que desea. En la tabla se indica Halo, Sample o Ambos (que significa que ambos temas admiten la propiedad de estilo).

El tema Halo es el tema predeterminado de los componentes. Por ello, si desea utilizar el tema Sample, deberá cambiar el tema actual de Halo a Sample.

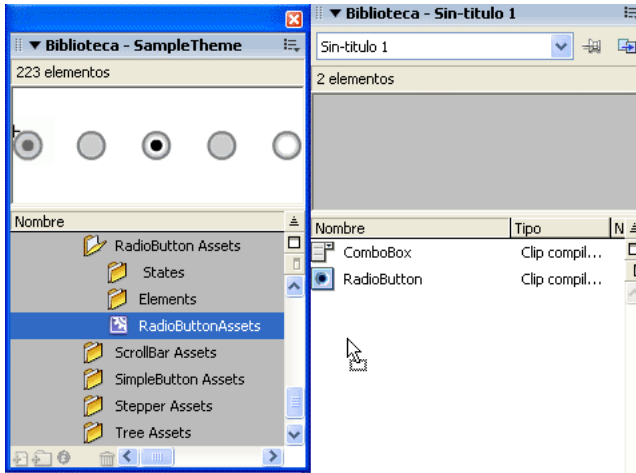
Para cambiar al tema Sample:

1. Seleccione Archivo > Abrir y abra el documento que utiliza componentes de la versión 2 en Flash o seleccione Archivo > Nuevo y cree un documento nuevo que utilice componentes de la versión 2.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione SampleTheme.fla para aplicarlo al documento.

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 la página 115.

3. En el panel Biblioteca del tema SampleTheme.fla, seleccione Flash UI Components 2/ Themes/MMDefault y arrastre la carpeta Assets de cualquier componente del documento al panel Biblioteca del documento de Flash.

Por ejemplo, arrastre la carpeta RadioButton Assets a la biblioteca.



Si no está seguro de qué componentes están en el documento, arrastre todo el clip de película del tema Sample al escenario. Los aspectos se asignan automáticamente a componentes del documento.

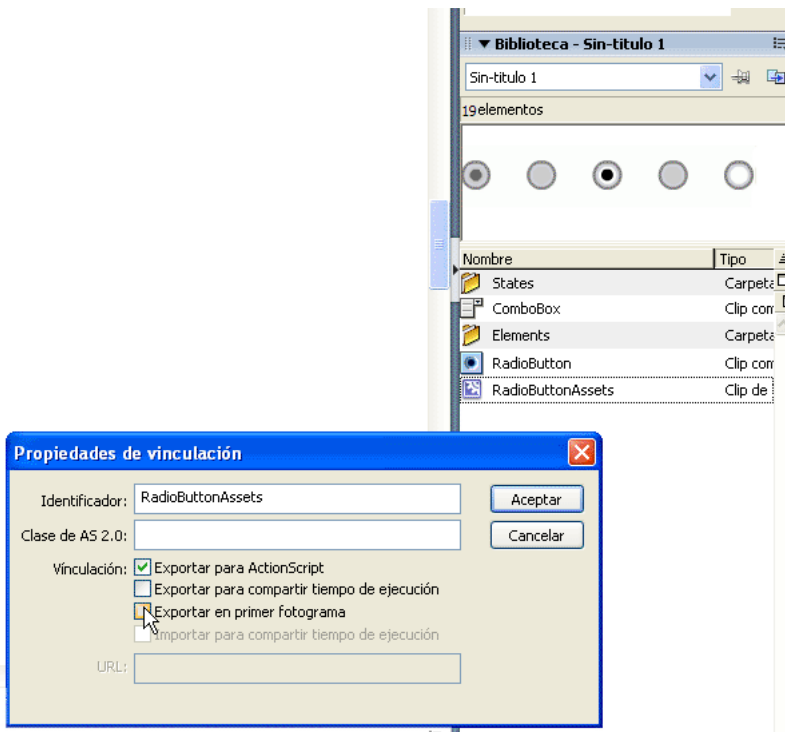
NOTA

La previsualización dinámica de los componentes del escenario no reflejará el tema nuevo.

4. Si arrastra carpetas Assets de componentes individuales al panel Biblioteca del documento, asegúrese de que el símbolo Assets de cada componente está establecido en Exportar en primer fotograma.

Por ejemplo, la carpeta Assets del componente RadioButton se denomina RadioButton Assets. Abra la carpeta RadioButtonAssets; verá un símbolo de clip de película denominado RadioButtonAssets. El símbolo RadioButtonAssets contiene todos los símbolos de activos individuales de este componente.

Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el símbolo RadioButtonAssets en la biblioteca del documento y seleccione la opción de menú Vinculación. Active la opción Exportar en primer fotograma para que todos los símbolos de activos individuales se exporten también en el primer fotograma. A continuación, haga clic en Aceptar para guardar la configuración.



5. Seleccione Control > Probar película para ver el documento con el tema nuevo aplicado.

Creación de un tema nuevo

Si no desea utilizar el tema Halo ni el tema Sample, puede modificar cualquiera de ellos para crear un tema nuevo.

Algunos aspectos de los temas tienen un tamaño fijo. Puede agrandarlos o reducirlos y los componentes cambiarán de tamaño automáticamente para coincidir con ellos. Otros aspectos están formados por varias piezas, algunas de ellas estáticas y otras no.

Algunos aspectos (por ejemplo `RectBorder` y `ButtonSkin`) utilizan la interfaz API de dibujo de `ActionScript` para dibujar sus gráficos, ya que es más eficiente desde el punto de vista del tamaño y del rendimiento. En dichos aspectos, puede utilizar código `ActionScript` como plantilla para ajustar los aspectos a sus necesidades.

Para ver una lista de los aspectos admitidos por cada componente y sus propiedades, consulte *Referencia del lenguaje de componentes*.

Para crear un tema nuevo:

1. Seleccione el archivo FLA del tema que desee utilizar como plantilla y haga una copia.
Asigne a la copia un nombre exclusivo como **MyTheme.fla**.
2. Seleccione Archivo > Abrir MyTheme.fla en Flash.
3. Seleccione Ventana > Biblioteca para abrir la biblioteca si aún no está abierta.
4. Haga doble clic en el símbolo de aspecto que desee modificar para abrirlo en el modo de edición de símbolos.

Los aspectos se encuentran en la carpeta Flash UI Componentes 2/Themes/MMDefault/*Component Assets* (en este ejemplo se usa *RadioButton Assets*).

5. Modifique el símbolo o elimine los gráficos y cree gráficos nuevos.
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
Por ejemplo, abra el elemento `States/RadioFalseDisabled` y cambie el color del círculo interior a gris claro.
6. Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información situada en la parte superior del escenario para volver al modo de edición de documentos.
7. Repita los pasos 4 a 6 hasta que haya terminado de editar todos los aspectos que desee cambiar.
8. Aplique `MyTheme.fla` a un documento siguiendo los pasos mostrados más adelante en este capítulo. (Véase [“Aplicación de un tema nuevo a un documento”](#) en la página 121.)

Modificación de los valores predeterminados de las propiedades de estilo de un tema

Los valores predeterminados de la propiedad de estilo proporcionados por cada tema se incluyen en una clase denominada Default. Para cambiar los valores predeterminados para un tema personalizado, cree una nueva clase de ActionScript denominada Default en un paquete apropiado para el tema, y realice los cambios que desee en la configuración predeterminada.

Para modificar los valores de estilo predeterminados de un tema:

1. Cree una nueva carpeta para el tema en First Run/Classes/mx/skins.
Por ejemplo, cree una carpeta denominada **myTheme**.
2. Copie una clase Defaults existente a la carpeta del nuevo tema.
Por ejemplo, copie `mx/skins/halo/Defaults.as` a `mx/skins/myTheme/Defaults.as`.
3. Abra la nueva clase Defaults en un editor de ActionScript.
Los usuarios de Flash Professional 8 pueden abrir el archivo en Flash. O bien, pueden abrirlo en el Bloc de notas de Windows o en SimpleText en Macintosh.
4. Modifique la declaración de clase para que refleje el nuevo paquete.
Por ejemplo, nuestra nueva declaración de clase es `class mx.skins.myTheme.Defaults`.
5. Realice los cambios que desee en la configuración del estilo.
Por ejemplo, cambie el color de estado desactivado a rojo oscuro.

```
o.disabledColor = 0x663333;
```
6. Guarde el archivo de la clase Defaults modificado.
7. Copie una clase FocusRect existente del tema de origen al tema personalizado.
Por ejemplo, copie `mx/skins/halo/FocusRect.as` a `mx/skins/myTheme/FocusRect.as`.
8. Abra la nueva clase FocusRect en un editor de ActionScript.
9. Cambie todas las referencias al paquete del tema de origen por referencias al paquete del nuevo tema.
Por ejemplo, cambie “halo” por “myTheme” todas las veces que aparezca.
10. Guarde el archivo de la clase FocusRect modificado.
11. Abra el archivo FLA para el tema personalizado.
En este ejemplo se usa `MyTheme fla`.
12. Abra la biblioteca (Window > Biblioteca) y busque el símbolo de Defaults.
En este ejemplo, está en `Flash UI Components 2/Themes/MMDefault/Defaults`.
13. Edite las propiedades del símbolo de Default.

14. Cambie la configuración de la clase de AS 2.0 para que refleje el nuevo paquete.
La clase de ejemplo es `mx.skins.myTheme.Defaults`.
15. Haga clic en Aceptar.
16. Busque el símbolo de FocusRect.
En este ejemplo, está en `Flash UI Components 2/Themes/MMDefault/FocusRect`.
17. Edite las propiedades del símbolo de FocusRect.
18. Cambie la configuración de la clase de AS 2.0 para que refleje el nuevo paquete.
La clase de ejemplo es `mx.skins.myTheme.FocusRect`.
19. Haga clic en Aceptar.
20. Aplique el tema personalizado a un documento mediante los pasos descritos en la siguiente sección.
No olvide incluir los símbolos de `Defaults` y `FocusRect` al arrastrar elementos del tema personalizado al documento de destino.

En este ejemplo ha usado un tema nuevo para personalizar el color del texto de componentes desactivados. Esta personalización específica (el cambio de un solo valor predeterminado de propiedad de estilo) se podría realizar de forma más sencilla mediante la aplicación de estilos, como se explica en [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 86](#). Es adecuado utilizar un tema nuevo para personalizar los valores predeterminados al personalizar muchas propiedades de estilo o al crear un tema nuevo para personalizar gráficos de componente.

Aplicación de un tema nuevo a un documento

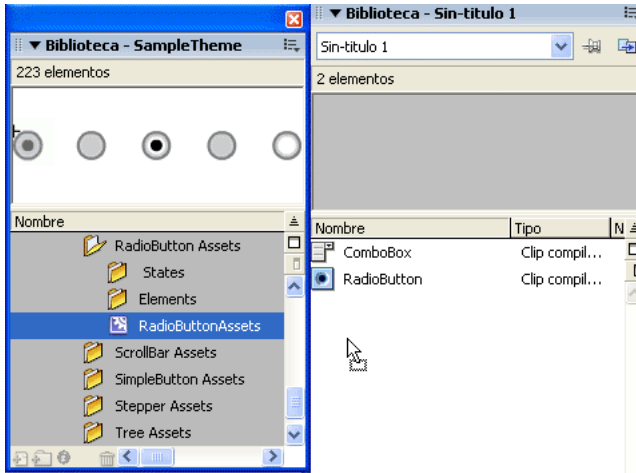
Para aplicar un tema nuevo a un documento, abra un archivo FLA del tema como biblioteca externa y arrastre la carpeta Theme desde la biblioteca externa hasta la biblioteca del documento. En los siguientes pasos se explica detalladamente el proceso, suponiendo que ya tiene un tema nuevo (para más información, consulte [“Creación de un tema nuevo” en la página 119](#)).

Para aplicar un tema a un documento:

1. Seleccione Archivo > Abrir y abra el documento que utiliza componentes de la versión 2 en Flash o seleccione Archivo > Nuevo y cree un documento nuevo que utilice componentes de la versión 2.
2. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo FLA del tema que desee aplicar al documento.

3. En el panel Biblioteca del tema, seleccione Flash UI Components 2/Themes/MMDefault y arrastre la carpeta Assets de cualquier componente que desee utilizar a la biblioteca del documento.

Por ejemplo, arrastre la carpeta RadioButton Assets a la biblioteca.



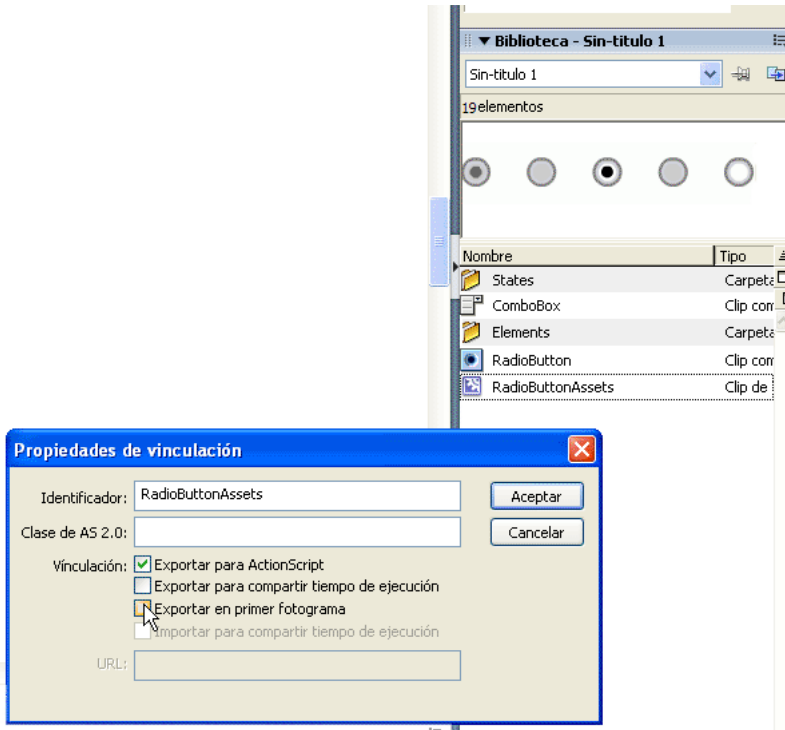
Si no está seguro de los componentes que están en el documento, arrastre todo el clip de película del tema al escenario (por ejemplo, para el archivo SampleTheme.fla, el clip de película del tema principal es Flash UI Components 2 > SampleTheme). Los aspectos se asignan automáticamente a componentes del documento.

NOTA

La previsualización dinámica de los componentes del escenario no reflejará el tema nuevo.

4. Si arrastró carpetas Assets individuales a la biblioteca ThemeApply.fla, asegúrese de que el símbolo Assets de cada componente está establecido en Exportar en primer fotograma.

Por ejemplo, la carpeta Assets para el componente RadioButton se denomina RadioButton Assets; tiene un símbolo denominado RadioButtonAssets, que contiene todos los símbolos de elementos individuales. Si establece Exportar en primer fotograma en el símbolo de RadioButtonAssets, todos los símbolos de activos individuales también se exportarán en el primer fotograma.



5. Seleccione Control > Probar película para el tema nuevo aplicado.

Cambio de la configuración de exportación

Cuando se aplica el tema Sample o Halo al documento, muchos de los aspectos se exportan en el primer fotograma para que los componentes puedan disponer de ellos de forma inmediata durante la reproducción. Sin embargo, si cambia la configuración de exportación de publicación (Archivo > Configuración de publicación > ficha Flash > botón Configuración de la versión de ActionScript > Fotograma de exportación para clases) del archivo FLA y establece un fotograma después del primero, también debe cambiar la configuración de exportación de los activos de los temas Sample y Halo. Para ello, debe abrir los siguientes activos del componente en la biblioteca del documento y desactivar la casilla de verificación Exportar en primer fotograma (haga clic con el botón derecho del ratón > Vinculación > Exportar en primer fotograma):

Tema Sample

- Flash UI Components 2/Base Classes/UIObject
- Flash UI Components 2/Themes/MMDefault/Defaults
- Flash UI Components 2/Base Classes/UIObjectExtensions
- Flash UI Components 2/Border Classes/BoundingBox
- Flash UI Components 2/SampleTheme
- Flash UI Components 2/Themes/MMDefault/Button Assets/Elements/ButtonIcon
- Flash UI Components 2/Themes/MMDefault/DateChooser Assets/ Elements/ Arrows/cal_disabledArrow
- Flash UI Components 2/Themes/MMDefault/FocusRect
- Flash UI Components 2/Themes/MMDefault/Window Assets/ States/ CloseButtonOver
- Flash UI Components 2/Themes/MMDefault/Accordion Assets/ AccordionHeaderSkin
- Flash UI Components 2/Themes/MMDefault/Alert Assets/AlertAssets
- Flash UI Components 2/Themes/MMDefault/Border Classes/Border
- Flash UI Components 2/Themes/MMDefault/Border Classes/CustomBorder
- Flash UI Components 2/Themes/MMDefault/Border Classes/RectBorder
- Flash UI Components 2/Themes/MMDefault/Button Assets/ActivatorSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin

Tema Halo

- Flash UI Components 2/Base Classes/UIObject
- Flash UI Components 2/Themes/MMDefault/Defaults
- Flash UI Components 2/Base Classes/UIObjectExtensions
- Flash UI Components 2/Component Assets/BoundingBox
- Flash UI Components 2/HaloTheme
- Flash UI Components 2/Themes/MMDefault/Accordion Assets/
AccordionHeaderSkin
- Flash UI Components 2/Themes/MMDefault/Alert Assets/AlertAssets
- Flash UI Components 2/Themes/MMDefault/Border Classes/Border
- Flash UI Components 2/Themes/MMDefault/Border Classes/CustomBorder
- Flash UI Components 2/Themes/MMDefault/Border Classes/RectBorder
- Flash UI Components 2/Themes/MMDefault/Button Assets/ActivatorSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/Elements/ButtonIcon
- Flash UI Components 2/Themes/MMDefault/CheckBox Assets/Elements/
CheckThemeColor1
- Flash UI Components 2/Themes/MMDefault/CheckBox Assets/CheckBoxAssets
- Flash UI Components 2/Themes/MMDefault/ComboBox Assets/ComboBoxAssets
- Flash UI Components 2/Themes/MMDefault/DataGrid Assets/DataGridAssets
- Flash UI Components 2/Themes/MMDefault/DateChooser Assets/
DateChooserAssets
- Flash UI Components 2/Themes/MMDefault/FocusRect
- Flash UI Components 2/Themes/MMDefault/Menu Assets/MenuAssets
- Flash UI Components 2/Themes/MMDefault/MenuBar Assets/MenuBarAssets
- Flash UI Components 2/Themes/MMDefault/ProgressBar Assets/ProgressBarAssets
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/Elements/
RadioThemeColor1
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/Elements/
RadioThemeColor2
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/
RadioButtonAssets
- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/HScrollBarAssets
- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/ScrollBarAssets

- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/VScrollBarAssets
- Flash UI Components 2/Themes/MMDefault/Stepper Assets/Elements/StepThemeColor1
- Flash UI Components 2/Themes/MMDefault/Stepper Assets/NumericStepperAssets
- Flash UI Components 2/Themes/MMDefault/Tree Assets/TreeAssets
- Flash UI Components 2/Themes/MMDefault/Window Assets/Window Assets

Combinación de aplicación de aspectos y estilos para personalizar un componente

En esta sección personalizará una instancia del componente ComboBox mediante la configuración de estilos, temas y aplicación de aspectos. En los procedimientos se muestra cómo combinar la configuración de aplicación de aspectos y estilos para crear una presentación única de un componente.

Creación de una instancia de componente en el escenario.

En la primera parte de este ejercicio, es preciso crear una instancia de ComboBox para personalizarla.

Para crear la instancia de ComboBox:

1. Arrastre un componente ComboBox al escenario.
2. En el panel Propiedades, asigne a la instancia el nombre `my_cb`.
3. En el primer fotograma de la línea de tiempo principal, añada el siguiente código ActionScript (asegúrese de que lo añada al fotograma y no al propio componente; en el panel Acciones debe leerse “Acciones - Fotograma” en la barra de título):


```
my_cb.addItem({data:1, label:"One"});
my_cb.addItem({data:2, label:"Two"});
```
4. Seleccione Control > Probar película para ver el cuadro combinado con el estilo y el aspecto predeterminado del tema Halo.

Creación de una nueva declaración de estilo

A continuación, debe crear una nueva declaración de estilo y asignarle estilos. Cuando la declaración de estilo contenga todos los estilos que desee, puede asignar el nuevo nombre de estilo a la instancia de ComboBox.

Para crear una nueva declaración de estilo y asignarle un nombre:

1. En el primer fotograma de la línea de tiempo principal, añada la siguiente línea al principio del código ActionScript (siguiendo una convención de programación, todas las sentencias de importación deben colocarse al principio del código ActionScript):

```
import mx.styles.CSSStyleDeclaration;
```

2. En la siguiente línea, asigne un nombre a la nueva declaración de estilo y añádala a las definiciones de estilos global:

```
var new_style:Object = new CSSStyleDeclaration();  
_global.styles.myStyle = new_style;
```

Después de asignar una nueva declaración de estilo a la hoja de estilos `_global`, puede asociar la configuración de estilo individual a la declaración de estilo `new_style`. Para más información sobre cómo crear una hoja de estilos para grupos de componentes en lugar de crear definiciones de estilos para una sola instancia, consulte [“Definición de estilos personalizados para grupos de componentes” en la página 92](#)).

3. Asocie algunas de las opciones de configuración de estilos a la declaración de estilo `new_style`. Las siguientes opciones de configuración de estilos incluyen las definiciones de estilos disponibles en el componente ComboBox (consulte [“Utilización de estilos con el componente ComboBox” en Referencia del lenguaje de componentes](#) para obtener una lista completa) además de los estilos de la clase `RectBorder`, dado que el componente `ComboBox` utiliza la clase `RectBorder`:

```
new_style.setStyle("textAlign", "right");  
new_style.setStyle("selectionColor", "white");  
new_style.setStyle("useRollOver", false);  
// borderStyle de la clase RectBorder  
new_style.setStyle("borderStyle", "none");
```

Asignación de definiciones de estilos al cuadro combinado

En este punto, tiene una declaración de estilo que contiene una variedad de estilos pero necesita asignar explícitamente el nombre de estilo a la instancia del componente. Puede asignar esta nueva declaración de estilo a *cualquier* instancia de componente en el documento, de la siguiente manera. Añada la siguiente línea después de las sentencias `addItem()` de `my_cb` (siguiendo una convención de programación, debería mantener juntas todas las sentencias de creación del cuadro combinado):

```
my_cb.setStyle("styleName", "myStyle");
```

El código `ActionScript` asociado al primer fotograma de la línea de tiempo principal debería ser el siguiente:

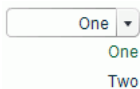
```
import mx.styles.CSSStyleDeclaration;

var new_style:Object = new CSSStyleDeclaration();
_global.styles.myStyle = new_style;

new_style.setStyle("textAlign", "right");
new_style.setStyle("selectionColor", "white");
new_style.setStyle("useRollOver", false);
// borderStyle de la clase RectBorder
new_style.setStyle("borderStyle", "none");

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

Seleccione `Control > Probar película` para ver el cuadro combinado con los estilos aplicados:



Cambio de tema del cuadro combinado

En cada componente de interfaz de usuario se enumeran las propiedades de estilo que pueden establecerse en dicho componente (por ejemplo, todas las propiedades de estilo que pueden establecerse para un componente ComboBox se enumeran en “Personalización del componente ComboBox” en *Referencia del lenguaje de componentes*). En la columna “Tema” de la tabla de propiedades de estilo se enumeran los temas instalados y las propiedades de estilo que admiten. No todos los temas instalados admiten todas las propiedades de estilo. El tema predeterminado de todos los componentes de interfaz de usuario es el tema Halo. Cuando cambie este tema por el tema Sample, puede utilizar un conjunto distinto de propiedades de estilo (es posible que algunas propiedades ya no estén disponibles si se enumeran solamente como tema Halo).

Para cambiar el tema del componente con estilos:

1. Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione SampleTheme.fla para abrir la biblioteca del tema Sample en Flash.

Este archivo se encuentra en la carpeta de configuración a nivel de la aplicación:

- En Windows: C:\Archivos de programa\Macromedia\Flex 8\idioma\Configuration\ComponentFLA\
- En Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/

2. Arrastre el clip de película principal SampleTheme (Flash UI Components 2 > SampleTheme) de la biblioteca de SampleTheme a la biblioteca del documento.

El componente ComboBox es una combinación de varios componentes y clases, y requiere activos de esos otros componentes y activos, incluidos Border y ScrollBar. Para asegurarse de que dispone de todos los activos necesarios de un tema, arrastre todos los activos del tema a la biblioteca.

3. Seleccione Control > Probar película para ver el cuadro combinado con los estilos aplicados:



Edición de activos de aspecto del cuadro combinado

Para editar la apariencia de un componente, edite gráficamente los aspectos que forman el componente. Para editar los aspectos, abra los activos gráficos del componente desde el tema actual y edite los símbolos de dicho componente. Macromedia recomienda seguir este procedimiento porque de esta forma no se eliminan ni se añaden símbolos que podrían necesitar otros componentes; con este procedimiento se edita la apariencia de un símbolo de aspecto del componente existente.

NOTA

Aunque no se recomienda, es *posible* editar los archivos de clase de origen de un componente para utilizar símbolos con distintos nombres como aspectos y modificar mediante programación el código ActionScript de un símbolo de aspecto (para obtener un ejemplo de código ActionScript y símbolos de aspecto personalizados, consulte “Personalización del componente Accordion (sólo en Flash Professional)” en *Referencia del lenguaje de componentes*). Sin embargo, dado que varios componentes, incluido el componente ComboBox, comparten activos, editar los archivos de origen o cambiar los nombres de símbolos de aspecto puede tener resultados inesperados.

Cuando se edita un símbolo de aspecto de un componente:

- Todas las instancias de dicho componente utilizarán nuevos aspectos (pero no estilos personalizados a menos que asocie explícitamente los estilos a las instancias) y algunos componentes dependientes de dicho componente utilizarán los nuevos aspectos.
- Si asigna un nuevo tema después de editar los aspectos del componente, asegúrese de no sobrescribir los aspectos “editados” existentes (en un cuadro de diálogo se le preguntará si desea sobrescribir los aspectos y podrá establecer que Flash deje de sobrescribir aspectos).

En esta sección, seguirá utilizando el cuadro combinado de la sección anterior (consulte “Cambio de tema del cuadro combinado” en la página 129). En los siguientes pasos se cambia por un círculo el aspecto de la flecha abajo que abre el menú de cuadro combinado.

Para editar el símbolo de flecha abajo de un cuadro combinado:

1. En la biblioteca del documento, abra los activos de ComboBox para ver los clips de película que son los aspectos del botón que abre y cierre la instancia de cuadro combinado en tiempo de ejecución. En concreto, abra la carpeta Themes > MMDefault > ComboBox Assets > States en la biblioteca del documento.

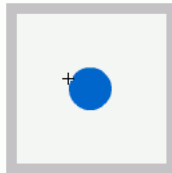
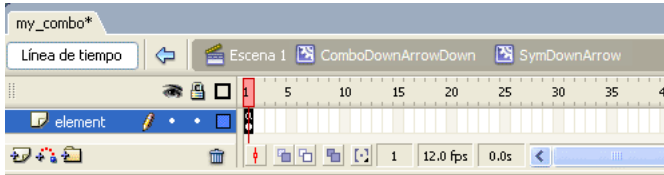
La carpeta States contiene cuatro clips de película: ComboDownArrowDisabled, ComboDownArrowDown, ComboDownArrowOver y ComboDownArrowUp. Estos cuatro símbolos se componen de otros símbolos. Y los cuatro utilizan el mismo símbolo para la flecha abajo (triángulo), denominado SymDownArrow.

2. Haga doble clic en el símbolo ComboDownArrowDown para editarlo.
Puede que necesite acercarlo hasta un 800% para ver los detalles del botón.

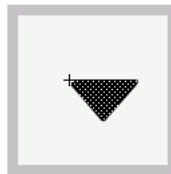
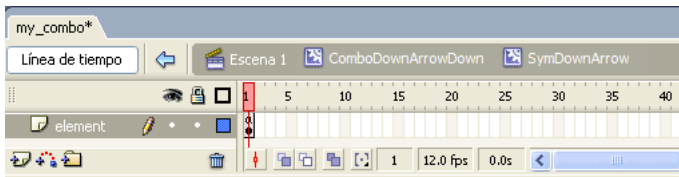
- Haga doble clic en la flecha abajo (triángulo negro) para editarlo.

NOTA

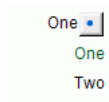
Asegúrese de que el símbolo SymDownArrow está seleccionado y que sólo está eliminando la forma del interior del clip de película y no el propio clip de película.



- Elimine la flecha abajo seleccionada (la forma de triángulo negro, *no todo el clip de película*) en el escenario.
- Mientras editar SymDownArrow, dibuje un círculo en el lugar donde estaba la flecha abajo. Para hacer más evidente el cambio, considere la posibilidad de dibujar un círculo de un color brillante, por ejemplo, azul, de aproximadamente 4 píxeles x 4 píxeles, y con una coordenada x establecida en 0 y una coordenada y establecida en -1 para que quede centrado.



6. Seleccione Control > Probar película para ver el cuadro combinado con el aspecto aplicado:



Si selecciona `ComboDownArrowOver` y `ComboDownArrowUp` en la biblioteca del documento, verá que también tienen un círculo azul en lugar de un triángulo negro, porque también utilizan `SymDownArrow` como símbolo de flecha abajo.

En este capítulo se describe cómo crear componentes propios y empaquetarlos para poder distribuirlos.

Este capítulo contiene las siguientes secciones:

Archivos de origen de componentes	133
Información general de la estructura de componentes	134
Creación del primer componente	136
Selección de una clase principal	146
Creación de un clip de película del componente	149
Creación del archivo de clase de ActionScript	154
Incorporación de componentes existentes en el componente	185
Exportación y distribución de un componente	195
Últimos pasos del desarrollo de componentes	199

Archivos de origen de componentes

Los componentes disponibles en el panel Componentes son clips SWC compilados previamente. El documento de Flash (FLA) de origen que contiene los gráficos y los archivos de clases de ActionScript (AS) de origen que contienen el código de estos componentes también se proporcionan para que pueda utilizarlos en la creación de sus propios componentes. Los archivos de origen de los componentes de la versión 2 se instalan con Macromedia Flash. Resulta útil abrir y revisar algunos de estos archivos e intentar comprender su estructura antes de crear sus propios componentes. Puede explorar, por ejemplo, el componente `RadioButton`, que es un buen ejemplo de componente sencillo. Todos los componentes son símbolos de la biblioteca de `StandardComponents fla`. Cada símbolo está vinculado a una clase de ActionScript. Su ubicación en la siguiente:

- Código fuente del archivo FLA
 - En Windows: C:\Archivos de programa\Macromedia\FIash 8\idioma\Configuration\ComponentFLA\StandardComponents fla.
 - En Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/StandardComponents fla
- Archivos de clase de ActionScript
 - En Windows: C:\Archivos de programa\Macromedia\FIash 8\idioma\First Run\Classes\mx
 - En Macintosh: Disco duro/Applications/Macromedia Flash 8/First Run/Classes/mx

Información general de la estructura de componentes

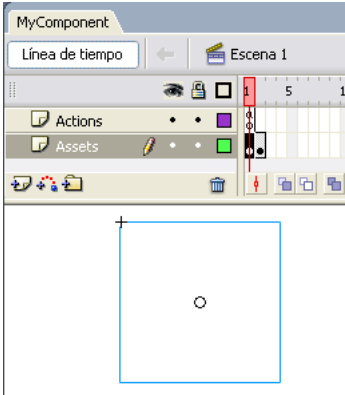
Un componente se compone de un archivo de Flash (FLA) y un archivo de ActionScript (AS). Hay otros archivos (por ejemplo, un icono y un archivo de depuración .swd) que puede crear opcionalmente y empaquetar con el componente, pero todos los componentes requieren un archivo FLA y un archivo de ActionScript. Cuando haya acabado de desarrollar el componente, expórtelo como un archivo SWC.



Un archivo de Flash (FLA), un archivo de ActionScript (AS) y un archivo de SWC

El archivo FLA contiene un símbolo de clip de película que debe vincularse al archivo AS en los cuadros de diálogo Propiedades de vinculación y Definición de componente.

El símbolo de clip de película tiene dos fotogramas y dos capas. La primera capa es una capa Acciones y contiene una función global `stop()` en el fotograma 1. La segunda capa es una capa Activos con dos fotogramas clave. El fotograma 1 contiene un cuadro de delimitación; el fotograma 2 contiene todos los demás activos, incluidos los gráficos y clases base que utiliza el componente.



El código ActionScript que especifica las propiedades y métodos del componente se encuentra en un archivo de clase de ActionScript independiente. Este archivo de clase también declara las clases que amplía el componente (si hay alguna). El nombre del archivo de clase AS es el nombre del componente, más la extensión “.as”. Por ejemplo, `MyComponent.as` contiene el código fuente del componente `MyComponent`.

Es aconsejable guardar los archivos FLA y AS del componente en la misma carpeta y con el mismo nombre. Si no se guarda el archivo AS en la misma carpeta, debe comprobar que la carpeta se encuentra en la ruta de clases para que el archivo FLA pueda encontrarlo. Para más información sobre la ruta de clases, consulte “Clases” en *Aprendizaje de ActionScript 2.0 en Flash*.

Creación del primer componente

En esta sección se creará un componente Dial. El conjunto de archivos del componente, Dial fla, Dial.as y DialAssets fla, se ubica en la carpeta de ejemplos del equipo:

- En Windows: carpeta C:\Archivos de programa\Macromedia\FIash8\Samples and Tutorials\Samples\Components\DialComponent.
- En Macintosh: carpeta Disco duro/Applications/Macromedia FIash 8/Samples and Tutorials/Samples/Components/DialComponent.

El componente Dial es un potenciómetro como los que se utilizan para medir la diferencia de tensión potencial. Un usuario puede hacer clic en el indicador y arrastrarlo para cambiar su posición. La interfaz API del componente Dial tiene una propiedad, `value`, que puede utilizar para obtener y establecer la posición del indicador.

En esta sección se describen los pasos para crear un componente. Estos procedimientos se tratan con más detalles en las siguientes secciones (“Selección de una clase principal” en la página 146, “Creación de un clip de película del componente” en la página 149, “Creación del archivo de clase de ActionScript” en la página 154 y “Exportación y distribución de un componente” en la página 195). Esta sección contiene los siguientes temas:

- “Creación del archivo Dial FIash (FLA)” en la página 136
- “Creación del archivo de clase Dial” en la página 140
- “Prueba y exportación del componente Dial” en la página 143

Creación del archivo Dial FIash (FLA)

El primer paso para crear un componente es crear el clip de película del componente en un archivo de documento de FLA.

Para crear el archivo Dial FLA:

1. En FIash, seleccione Archivo > Nuevo y cree un nuevo documento.
2. Seleccione Archivo > Guardar como y guarde el archivo como **Dial fla**.
El archivo puede tener cualquier nombre pero resulta práctico asignarle el mismo nombre que el componente.
3. Seleccione Insertar > Nuevo símbolo. El propio componente se crea como un nuevo símbolo MovieClip que estará disponible en toda la biblioteca.
Denomine al componente Dial y asígnele el comportamiento Clip de película.
4. Si la sección Vinculación del cuadro de diálogo Crear un nuevo símbolo no se abre, haga clic en el botón Avanzado para mostrarla.

5. En el área Vinculación, active la opción Exportar para ActionScript y desactive Exportar en primer fotograma.
6. En el cuadro de texto Identificador, introduzca un identificador de vinculación como **Dial_ID**.
7. En el cuadro de texto Clase de AS 2.0, introduzca **Dial**. Este valor es el nombre de la clase del componente. Si la clase está contenida en un paquete (por ejemplo, `mx.controls.Button`), introduzca el nombre completo del paquete.
8. Haga clic en Aceptar.
Flash cambia al modo de edición de símbolos.
9. Inserte una nueva capa. Denomine a la capa superior **Acciones** y a la capa inferior **Activos**.
10. Seleccione el fotograma 2 de la capa Activos e inserte un fotograma clave (F6).
La estructura del clip de película del componente consta de dos elementos: una capa Acciones y una capa Activos. La capa Acciones tiene un fotograma clave y la capa Activos tiene dos fotogramas clave.
11. Seleccione el fotograma 1 de la capa Acciones y abra el panel Acciones (F9). Introduzca una función global `stop()`.
De esta forma evitará que el clip de película continúe en el fotograma 2.
12. Seleccione Archivo > Importar > Abrir biblioteca externa y elija el archivo `StandardComponents.fla` de la carpeta `Configuration/ComponentFLA`. Por ejemplo:
 - En Windows: `C:\Archivos de programa\Macromedia\Flash 8\idioma\Configuration\ComponentFLA\StandardComponents.fla`.
 - En Macintosh: `Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/StandardComponents.fla`

NOTA

Para más información sobre las ubicaciones de carpeta, consulte “Carpetas de configuración instaladas con Flash” en *Primeros pasos con Flash*.

13. Dial amplía la clase base `UIComponent`; por tanto, debe arrastrar una instancia de `UIComponent` al documento Dial. En la biblioteca `StandardComponents.fla`, busque el clip de película `UIComponent` en la siguiente carpeta: `Flash UI Components 2 > Base Classes > FUIObject Subclasses` y arrástrelo a la biblioteca `Dial.fla`.

Las dependencias de activos se copian automáticamente a la biblioteca Dial con `UIComponent`.

NOTA

Al arrastrar `UIComponent` a la biblioteca Dial, cambia la jerarquía de carpetas de la biblioteca Dial. Si tiene previsto utilizar nuevamente la biblioteca o usarla con otros grupos de componentes (por ejemplo, de la versión 2), debería estructurar de nuevo la jerarquía de carpetas para que coincida con la biblioteca `StandardComponents.fla` y, de este modo, esté bien organizada y evite la duplicación de símbolos.

14. En la capa `Activos`, seleccione el fotograma 2 y arrastre una instancia de `UIComponent` al escenario.
15. Cierre la biblioteca `StandardComponents.fla`.
16. Seleccione `Archivo > Importar > Abrir biblioteca externa` y seleccione el archivo `DialAssets.fla`.
 - En Windows: `C:\Archivos de programa\Macromedia\Flesh8\Samples and Tutorials\Samples\Components\DialComponent\DialAssets.fla`.
 - En Macintosh: `Disco duro/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/DialComponent/DialAssets.fla`
17. En la capa `Activos`, seleccione el fotograma 2 y arrastre una instancia del clip de película `DialFinal` de la biblioteca `DialAssets` al escenario.

Todos los activos del componente se añaden al fotograma 2 de la capa `Activos`. Como hay una función global `stop()` en el fotograma 1 de la capa `Acciones`, los activos del fotograma 2 no se verán mientras se organizan en el escenario.

Se añaden activos al fotograma 2 por dos motivos:

- Para que todos los activos y subactivos se copien automáticamente a la biblioteca y estén disponibles para crear dinámicamente una instancia (en el caso de `DialFinal`) o acceder a sus métodos, propiedades y eventos (en el caso de `UIComponent`).
 - Para asegurarse, mediante la colocación de activos en un fotograma, de que se cargan sin problemas mientras se reproduce la película, evitando tener que establecer la exportación de activos de la biblioteca en el primer fotograma. Este método evita un pico inicial de transferencia de datos durante la descarga.
18. Cierre la biblioteca `DialAssets.fla`.

19. Seleccione el fotograma 1 de la capa Activos. Arrastre el clip de película BoundingBox de la biblioteca (carpeta Flash UI Components 2 > Component Assets) al escenario. Asigne a BoundingBox el nombre de instancia **boundingBox_mc**. Utilice el panel Información para establecer la altura y anchura del clip de película DialFinal en **250** píxeles, y las coordenadas x, y en **0, 0**.

La instancia de BoundingBox se utiliza para crear la previsualización dinámica del componente y cambiar el tamaño durante la edición. Debe establecer el tamaño del cuadro de delimitación para que pueda contener todos los elementos gráficos del componente.

NOTA

Si amplía un componente (incluido cualquier componente de la versión 2), debe mantener los nombres de instancia que ya utilice ese componente, ya que su código hará referencia a dichos nombres de instancia. Por ejemplo, si incluye un componente de la versión 2 que ya utilice el nombre de instancia `boundingBox_mc`, no debe cambiar el nombre. En los nombres de instancia que cree, puede utilizar cualquier nombre exclusivo que no entre en conflicto con ninguno de los que ya existen en el mismo ámbito.

20. Seleccione el clip de película Dial en la biblioteca y elija Definición de componente en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
21. En el cuadro de texto Clase de AS 2.0, introduzca **Dial**.
Este valor es el nombre de la clase de ActionScript. Si la clase está en un paquete, el valor es el paquete completo, por ejemplo, `mx.controls.CheckBox`.
22. Haga clic en Aceptar.
23. Guarde el archivo.

Creación del archivo de clase Dial

A continuación, necesita crear el archivo de clase Dial como un nuevo archivo de ActionScript.

Para crear el archivo de clase Dial:

1. En Flash, seleccione Archivo > Nuevo y, a continuación, seleccione Archivo ActionScript.
2. Seleccione Archivo > Guardar como y guarde el archivo como **Dial.as** en la misma carpeta que el archivo Dial.fla.

NOTA

Puede utilizar cualquier editor de texto para guardar el archivo Dial.as.

3. Puede copiar o escribir el siguiente código de la clase de ActionScript del componente Dial en su nuevo archivo Dial.as. Si escribe el código en lugar de copiarlo, podrá familiarizarse con cada elemento del código de componente.

Lea los comentarios del código para obtener una descripción de cada sección. Para más información sobre los elementos de un archivo de clase de componente, consulte [“Información general sobre el archivo de clase de un componente” en la página 156](#).

```
// Importar el paquete para poder hacer referencia a
// la clase directamente.
import mx.core.UIComponent;

// Etiqueta de metadatos Event
[Event("change")]
class Dial extends UIComponent
{
    // Los componentes deben declarar esto para ser
    // componentes adecuados del marco de componentes.
    static var symbolName:String = "Dial";
    static var symbolOwner:Object = Dial;
    var className:String = "Dial";

    // Los clips de película del indicador y el dial que son
    // la representación gráfica del componente
    private var needle:MovieClip;
    private var dial:MovieClip;
    private var boundingBox_mc:MovieClip;

    // La variable de miembro privada "__value" es accesible
    // públicamente mediante métodos de captador/definidor implícitos
    // Al actualizar esta propiedad, se actualiza la posición del
    // indicador
    // cuando se establece el valor.
    private var __value:Number = 0;
```

```

// Esta marca se establece cuando el usuario arrastra
// el indicador con el ratón y se borra después.
private var dragging:Boolean = false;

// Constructor;
// Necesario para todas las clases; en componentes de la versión 2
// el constructor debe tener cero argumentos.
// Toda la inicialización se produce en un método init()
// necesario después de construir la instancia de la clase.
function Dial() {
}

// Código de inicialización:
// El método init() es necesario en componentes de la versión 2. Debe
// llamar también
// al método init() de su clase principal con super.init().
// El método init() es necesario en componentes que amplían
// UIComponent.
function init():Void {
    super.init();
    useHandCursor = false;
    boundingBox_mc._visible = false;
    boundingBox_mc._width = 0;
    boundingBox_mc._height = 0;
}
// Crear objetos secundarios necesarios al iniciar:
// El método createChildren() es necesario en componentes
// que amplían UIComponent.
public function createChildren():Void {
    dial = createObject("DialFinal", "dial", 10);
    size();
}

// El método draw() es necesario en componentes de la versión 2.
// Se invoca tras invalidar el componente
// mediante una llamada a invalidate().
// Es mejor que volver a dibujar desde la función set()
// del valor porque si hay otras propiedades, es
// mejor agrupar los cambios en un nuevo dibujo que
// hacerlos individualmente. Este enfoque permite
// centralizar más y mejor el código.
function draw():Void {
    super.draw();
    dial.needle._rotation = value;
}

// El método size() se invoca cuando cambia el tamaño del
// componente. Es una oportunidad para cambiar el tamaño de elementos
// secundarios,

```

```

// y de los gráficos del dial y del indicador.
// El método size() es necesario en componentes que amplían
// UIComponent.
function size():Void {
    super.size();
    dial._width = width;
    dial._height = height;
    // Vuelve a dibujar el indicador en caso necesario.
    invalidate();
}

// Es la función de captador/definidor de la propiedad value.
// Los metadatos [Inspectable] hacen que la propiedad aparezca
// en el inspector de propiedades. Es una función de captador/
// definidor,
// por lo que puede invalidarse la llamada y obligar a que
// se redibuje el componente cuando se cambia el valor.
[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue=0)]
function set value (val:Number)
{
    __value = val;
    invalidate();
}

function get value ():Number
{
    return twoDigits(__value);
}

function twoDigits(x:Number):Number
{
    return (Math.round(x * 100) / 100);
}

// Indica al componente que debe esperar que se presione el ratón
function onPress()
{
    beginDrag();
}

// Cuando se presiona el dial, se establece el indicador de arrastre.
// Se asignan funciones callback a los eventos del ratón.
function beginDrag()
{
    dragging = true;
    onMouseMove = mouseMoveHandler;
    onMouseUp = mouseUpHandler;
}

```

```

    }

    // Eliminar los eventos del ratón al finalizar el arrastre
    // y borrar el indicador.
    function mouseUpHandler()
    {
        dragging = false;
        delete onMouseMove;
        delete onMouseUp;
    }

    function mouseMoveHandler()
    {
        // Calcular el ángulo
        if (dragging) {
            var x:Number = _xmouse - width/2;
            var y:Number = _ymouse - height/2;

            var oldValue:Number = value;
            var newValue:Number = 90+180/Math.PI*Math.atan2(y, x);
            if (newValue<0) {
                newValue += 360;
            }
            if (oldValue != newValue) {
                value = newValue;
                dispatchEvent( {type:"change"} );
            }
        }
    }
}

```

Prueba y exportación del componente Dial

Ha creado el archivo de Flash que contiene los elementos gráficos y las clases base, y el archivo de clase que contiene toda la funcionalidad del componente Dial. Es el momento de probar el componente.

Lo ideal sería ir probando el componente mientras se trabaja, especialmente mientras se escribe el archivo de clase. La forma más rápida de realizar pruebas mientras se trabaja es convertir el componente en un clip compilado y utilizarlo en el archivo FLA del componente.

Cuando haya finalizado el componente, expórtelo como un archivo SWC. Para más información, consulte [“Exportación y distribución de un componente” en la página 195](#).

Para probar el componente Dial:

1. En el archivo Dial.fla, seleccione el componente Dial, abra el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada) y seleccione Convertir en clip compilado.

Se añadirá a la biblioteca un clip compilado denominado Dial SWF.

NOTA

Si ya ha creado un clip compilado (por ejemplo, si es la segunda o tercera vez que realiza pruebas), aparecerá un cuadro de diálogo Solucionar conflicto de biblioteca. Seleccione Reemplazar elementos existentes para añadir la nueva versión del documento.

2. Arrastre Dial SWF al escenario en la línea de tiempo principal.
3. Puede cambiar su tamaño y establecer su propiedad value en el inspector de propiedades o el inspector de componentes. Cuando establezca su propiedad value, la posición del indicador debería cambiar de forma correspondiente.
4. Para probar la propiedad value en tiempo de ejecución, asigne al dial el nombre de instancia **dial** y añada el siguiente código al fotograma 1 en la línea de tiempo principal:

```
// posición del campo de texto
var textXPos:Number = dial.width/2 + dial.x
var textYPos:Number = dial.height/2 + dial.y;

// crea un campo de texto donde puede verse dial.value
createTextField("dialValue", 10, textXPos, textYPos, 100, 20);

// crea un detector para gestionar el evento change
function change(evt){
// coloca la propiedad value en el campo de texto
// cuando se mueve el indicador
    dialValue.text = dial.value;
}
dial.addEventListener("change", this);
```

5. Seleccione Control > Probar película para probar el componente en Flash Player.

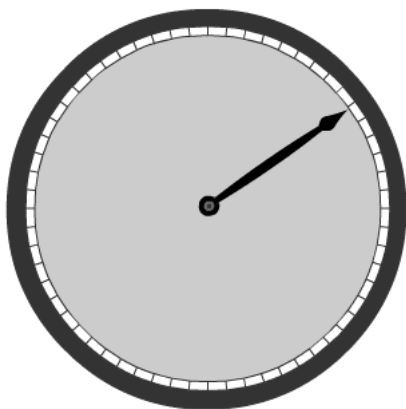
Para exportar el componente Dial:

1. En el archivo Dial fla, seleccione el componente Dial, abra el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada) y seleccione Exportar archivo SWC.
2. Seleccione la ubicación donde se guardará el archivo SWC.

Si se guarda en la carpeta Componentes de la carpeta de configuración de usuario, puede volver a cargar el panel Componentes sin reiniciar Flash y el componente aparecerá en el panel.

NOTA

Para más información sobre las ubicaciones de carpeta, consulte “Carpetas de configuración instaladas con Flash” en *Utilización de Flash*.



El componente Dial completo

Selección de una clase principal

Lo primero que hay que decidir a lo hora de crear un componente es si debe ampliarse una de las clases de la versión 2. Si elige ampliar una clase de la versión 2, puede ampliar una clase de componente (Button, CheckBox, ComboBox, List, etc.) o una de las clase base, UIObject o UICComponent. Todas las clases de componente, excepto de los componentes multimedia, amplían las clases base; si se amplía una clase de componente, la clase automáticamente recibe herencia de las clases base también.

Las dos clases base proporcionan funciones comunes para los componentes. Si se amplían estas clases, el componente empieza con un conjunto básico de métodos, propiedades y eventos.

No es necesario crear una subclase UIObject o UICComponent, ni cualquier otra clase del marco de la versión 2. Aunque las clases de componente hereden directamente de la clase MovieClip, puede utilizar muchas funciones eficaces de componente: exportar a un archivo SWC o clip compilado, utilizar una previsualización dinámica incorporada, ver propiedades de tipo inspectable, etc. Sin embargo, si desea que los componentes funcionen con los componentes de la versión 2 de Macromedia y desea utilizar las clases de administrador, es necesario ampliar UIObject o UICComponent.

En la tabla siguiente se describen brevemente las clases base de la versión 2:

Clase base	Amplía	Descripción
<code>mx.core.UIObject</code>	MovieClip	UIObject es la clase base para todos los objetos gráficos. Puede tener forma, dibujarse a sí misma y ser invisible. UIObject proporciona las funciones siguientes: <ul style="list-style-type: none">• Edición de estilos• Gestión de eventos• Cambio del tamaño mediante la escala
<code>mx.core.UICComponent</code>	UIObject	UICComponent es la clase base para todos los componentes. UICComponent proporciona las funciones siguientes: <ul style="list-style-type: none">• Creación de desplazamientos de la selección• Creación de un esquema de tabulación• Activación y desactivación de componentes• Cambio de tamaño de los componentes• Gestión de eventos de teclado y ratón de bajo nivel

Aspectos básicos de la clase UIObject

Los componentes que se basan en la versión 2 de la arquitectura de componentes de Macromedia proceden de la clase UIObject, que es una subclase de la clase MovieClip. La clase MovieClip es la clase base de todas las clases de Flash que representan objetos visuales en la pantalla.

UIObject añade métodos que permiten gestionar estilos y eventos. Envía eventos a sus detectores justo antes de dibujar (el evento `draw` es el equivalente del evento `MovieClip.onEnterFrame`), cuando se carga y descarga (`load` y `unload`), cuando su diseño cambia (`move`, `resize`) y cuando se oculta o se muestra (`hide` y `reveal`).

UIObject proporciona variables de sólo lectura alternativas para determinar la posición y el tamaño de un componente (`width`, `height`, `x`, `y`) y los métodos `move()` y `setSize()` para modificar la posición y el tamaño de un objeto.

La [Clase UIObject](#) implementa lo siguiente:

- Estilos
- Eventos
- Cambio del tamaño mediante la escala

Aspectos básicos de la clase UIComponent

La clase UIComponent es una subclase de UIObject (consulte [Clase UIComponent](#) en *Referencia del lenguaje de componentes*). Es la clase base de todos los componentes que gestionan la interacción con el usuario (entradas de teclado y de ratón). La clase UIComponent permite a los componentes realizar lo siguiente:

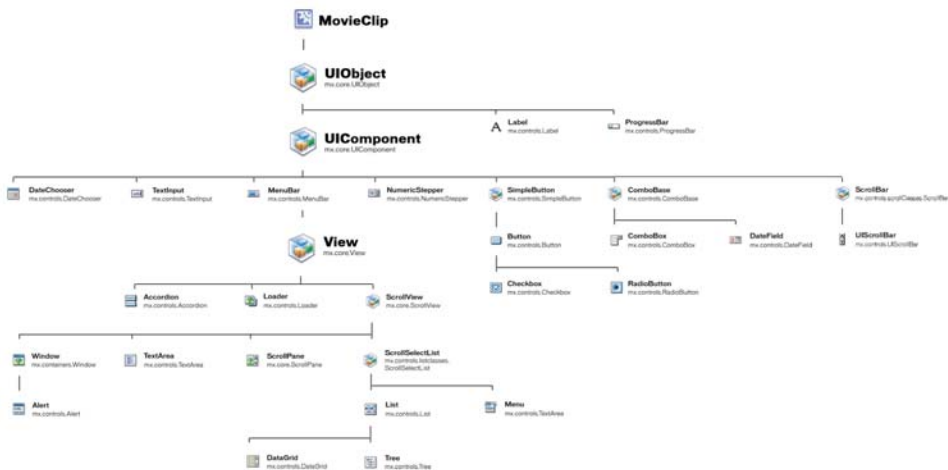
- Recibir selecciones y entradas del teclado
- Activar y desactivar componentes
- Cambiar el tamaño mediante el diseño

Ampliación de otras clases de la versión 2

Para facilitar la construcción de componentes, puede ampliar cualquier clase; no es obligatorio ampliar las clases `UIObject` o `UIComponent` directamente. Si amplía cualquier otra clase de componentes de la versión 2 (excepto de los componentes `Media`), se amplían de forma predeterminada `UIObject` y `UIComponent`. Cualquier clase de componente que aparezca en el diccionario de componentes puede ampliarse para crear una nueva clase de componente.

Por ejemplo, si desea crear un componente que se comporte de forma casi exacta que el componente `Button`, puede ampliar la clase `Button` en lugar de volver a crear todas las funciones de la clase `Button` desde las clases base.

La siguiente ilustración muestra la jerarquía de componentes de la versión 2:



Jerarquía de componentes de la versión 2

Hay una versión en FlashPaper de este archivo disponible en el directorio de instalación de Flash, en esta ubicación: `Flash 8\Samples and Tutorials\Samples\Components\arch_diagram.swf`.

Ampliación de la clase `MovieClip`

Puede elegir no ampliar una clase de la versión 2 y hacer que el componente herede directamente de la clase `MovieClip` de `ActionScript`. Sin embargo, si desea obtener la funcionalidad de `UIObject` y `UIComponent`, deberá crearla. Puede abrir las clases `UIObject` y `UIComponent` (`First Run/Classes/mx/core`) para examinar cómo se crearon.

Creación de un clip de película del componente

Para crear un componente, es necesario crear un símbolo de clip de película y vincularlo al archivo de clase del componente.

El clip de película tiene dos fotogramas y dos capas. La primera capa es una capa Acciones y contiene una función global `stop()` en el fotograma 1. La segunda capa es una capa Activos con dos fotogramas clave. El fotograma 1 contiene un cuadro de delimitación o cualquier gráfico que sirva de marcador de posición para la ilustración final. El fotograma 2 contiene todos los demás activos, incluidos los gráficos y clases base que utiliza el componente.

Inserción de un nuevo símbolo de clip de película

Todos los componentes son objetos `MovieClip`. Para crear un nuevo componente, primero debe insertar un nuevo símbolo en un nuevo archivo FLA.

Para añadir el símbolo de un nuevo componente:

1. En Flash, cree un documento de Flash vacío.
2. Seleccione Insertar > Nuevo símbolo.
Aparecerá el cuadro de diálogo Crear un nuevo símbolo.
3. Especifique un nombre de símbolo. Asigne un nombre al componente poniendo en mayúscula la primera letra de cada una de las palabras del componente (por ejemplo, `MyComponent`).
4. Seleccione un comportamiento Clip de película.
5. Haga clic en el botón Avanzado para mostrar la configuración avanzada.
6. Active la opción Exportar para ActionScript y desactive Exportar en primer fotograma.
7. Introduzca un identificador de vinculación.

8. En el cuadro de texto Clase de AS 2.0, introduzca la ruta completa a la clase de ActionScript 2.0.

El nombre de clase debe coincidir con el nombre de componente que aparece en el panel Componentes. Por ejemplo, la clase del componente Button es `mx.controls.Button`.

NOTA

No incluya la extensión del nombre de archivo; el cuadro de texto Clase de AS 2.0 apunta a la ubicación del paquete de la clase y no al nombre que ha otorgado al archivo el sistema de archivos.

Si el archivo de ActionScript se encuentra en un paquete, debe incluir el nombre del paquete. Este valor puede ser relativo a la ruta de clases o también una ruta de paquetes absoluta (por ejemplo, `mypackage.MyComponent`).

9. En la mayoría de los casos, debe anular la selección de Exportar en primer fotograma (está seleccionada de forma predeterminada). Para más información, consulte [“Lista de comprobación de desarrollo de componentes” en la página 200](#).
10. Haga clic en Aceptar.
Flash añade el símbolo a la biblioteca y cambia al modo de edición de símbolos. En este modo, el nombre del símbolo aparece encima de la esquina superior izquierda del escenario y una cruz indica el punto de registro del símbolo.

Edición del clip de película

Una vez haya creado el nuevo símbolo y definido sus vínculos, puede definir los activos del componente en la línea de tiempo del símbolo.

El símbolo de un componente debe tener dos capas. En esta sección se describe qué capas deben insertarse y qué debe añadirse a estas capas.

Para editar el clip de película:

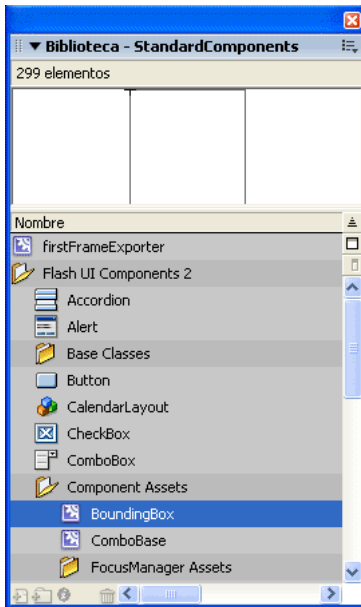
1. Cambie el nombre de la capa 1 **Acciones** y seleccione el fotograma 1.
2. Abra el panel Acciones y añada una función `stop()`, como se muestra a continuación:

```
stop();
```

No añada ningún activo gráfico a este fotograma.
3. Añada una capa denominada **Activos**.
4. Seleccione el fotograma 2 de la capa Activos e inserte un fotograma clave en blanco.
Esta capa contiene ahora dos fotogramas clave en blanco.

5. Siga uno de estos procedimientos:

- Si el componente tiene activos visuales que definen el área de delimitación, arrastre los símbolos al fotograma 1 y organícelos de la forma adecuada.
- Si el componente crea todos los elementos visuales en tiempo de ejecución, arrastre un símbolo BoundingBox al escenario en el fotograma 1, cambie su tamaño adecuadamente y asigne a la instancia el nombre **boundingBox_mc**. El símbolo se encuentra en la biblioteca de StandardComponents.fla, que se encuentra en la carpeta Configuration/ComponentFLA.



6. Si amplía un componente existente, coloque una instancia de dicho componente y cualquier otra clase base en el fotograma 2 de la capa Activos.

Para ello, seleccione el símbolo del panel Componentes y arrástrelo al escenario. Si amplía una clase base, abra StandardComponents.fla en la carpeta Configuration/ComponentFLA y arrastre la clase desde la biblioteca al escenario.

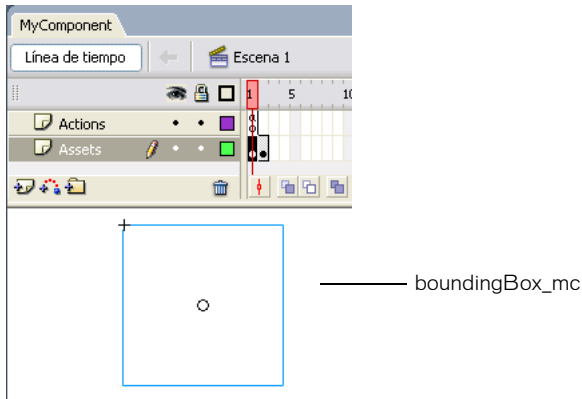
NOTA

Al arrastrar UICComponent a la biblioteca de componentes, cambia la jerarquía de carpetas de la biblioteca. Si tiene previsto utilizar nuevamente la biblioteca o usarla con otros grupos de componentes (por ejemplo, de la versión 2), debería estructurar de nuevo la jerarquía de carpetas para que coincida con la biblioteca StandardComponents.fla y, de este modo, esté bien organizada y evite la duplicación de símbolos.

7. Añada los activos gráficos que utilice este componente en el fotograma 2 de la capa Activos de su componente.

Todos los activos que utilice el componente (ya sea otro componente o medios como mapas de bits) deben tener una instancia en el segundo fotograma de la capa Activos.

8. El símbolo finalizado debería tener el siguiente aspecto:



Definición del clip de película como un componente

El símbolo de clip de película debe estar vinculado a un archivo de clase de ActionScript en el cuadro de diálogo Definición de componente. De esta forma, Flash sabe dónde debe buscar las metaetiquetas del componente. Para más información sobre metaetiquetas, consulte [“Adición de metadatos de componente” en la página 161](#). El cuadro de diálogo Definición de componente contiene otras opciones que también puede elegir.

Para definir un clip de película como un componente:

1. Seleccione el clip de película en la biblioteca y elija Definición de componente en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
2. Debe introducir una clase de AS 2.0.

Si la clase forma parte del paquete, introduzca el nombre completo del paquete.

3. Si lo desea, especifique otras opciones del cuadro de diálogo Definición de componente:
- Haga clic en el botón Más (+) para definir parámetros.
Esto es opcional. Se aconseja utilizar la etiqueta Inspectable de metadatos en el archivo de clase del componente para especificar los parámetros. Si no se especifica una clase de ActionScript 2.0, defina aquí los parámetros del componente.
 - Especifique una interfaz de usuario personalizada.
Es un archivo SWF que se reproduce en el inspector de componentes. Puede incorporarlo en el archivo FLA del componente o navegar hasta un archivo SWF externo.
 - Especifique una previsualización dinámica.
Es un archivo SWF incorporado o externo. No es necesario especificar aquí una previsualización dinámica; puede añadir un cuadro de delimitación en el clip de película del componente y Flash crea una previsualización dinámica. Véase [“Creación de un clip de película del componente” en la página 149](#).
 - Introduzca una descripción.
El campo Descripción está desfasado en Flash MX 2004 porque se ha eliminado el panel Referencia. Este campo se ofrece para proporcionar compatibilidad con versiones anteriores cuando se guardan archivos FLA en el formato de Flash MX.
 - Seleccione un icono.
Esta opción especifica un archivo PNG que se utilizará como icono para el componente. Si se especifica la etiqueta de metadatos IconFile en el archivo de clase de ActionScript 2.0 (es la práctica recomendada), se omite este campo.
 - Active o desactive la opción Los parámetros están bloqueados en instancias.
Cuando esta opción está desactivada, los usuarios pueden añadir parámetros a cada una de las instancias del componente que son distintas de los parámetros del componente. Normalmente, esta opción debería estar activada. Esta opción proporciona compatibilidad con Flash MX.
 - Especifique una sugerencia que aparezca en el panel Componentes.

Creación del archivo de clase de ActionScript

Todos los símbolos de componentes se vinculan a un archivo de clase de ActionScript 2.0. Para más información sobre vinculación, consulte [“Creación de un clip de película del componente” en la página 149](#).

Para editar los archivos de clase de ActionScript, puede utilizar Flash, cualquier editor de texto o cualquier entorno de desarrollo integrado (IDE, Integrated Development Environment).

La clase de ActionScript externa amplía otra clase (tanto si es una clase de un componente de la versión 2, una clase base de la versión 2 o la clase MovieClip de ActionScript). Debería ampliar la clase que crea la funcionalidad que más se asemeja al componente que desea crear. Sólo puede heredar de (ampliar) una clase. ActionScript 2.0 no permite la herencia múltiple.

Ejemplo sencillo de un archivo de clase de componente

A continuación se muestra un ejemplo sencillo de un archivo de clase denominado MyComponent.as. Si creara este componente, vincularía este archivo al clip de película del componente en Flash.

Este ejemplo contiene un conjunto mínimo de importaciones, métodos y declaraciones de un componente, MyComponent, que hereda de la clase UIComponent. El archivo MyComponents.as se guarda en la carpeta myPackage.

```
[Event("eventName")]

// Importar paquetes.
import mx.core.UIObject;

// Declarar la clase y ampliarla desde la clase principal.
class mypackage.MyComponent extends UIObject {

    // Identificar el nombre de símbolo vinculado a esta clase.
    static var symbolName:String = "mypackage.MyComponent";

    // Identificar el nombre de paquete completo del propietario del símbolo.
    static var symbolOwner:Object = Object(mypackage.MyComponent);

    // Proporcionar la variable className.
    var className:String = "MyComponent";

    // Definir un constructor vacío.
    function MyComponent() {
```

```

}

// Llamar al método init() del elemento principal.
// Ocultar el cuadro de delimitación; se utiliza
// sólo durante la edición.
function init():Void {
    super.init();

    boundingBox_mc.width = 0;
    boundingBox_mc.height = 0;
    boundingBox_mc.visible = false;
}

function createChildren():Void{
    // Llamar a createClassObject para crear subobjetos.
    size();
    invalidate();
}

function size(){
    // Escribir código para gestionar el cambio de tamaño.
    super.size();
    invalidate();
}

function draw(){
    // Escribir código para gestionar la representación visual.
    super.draw();
}
}

```

Información general sobre el archivo de clase de un componente

El siguiente procedimiento general describe cómo crear un archivo de ActionScript para una clase de componente. Algunos de los pasos pueden ser opcionales, en función del tipo de componente que haya creado.

Para escribir un archivo de clase del componente:

1. (Opcional) Importe las clases. (Véase [“Importación de clases” en la página 157.](#))
De esta forma, podrá hacer referencia a las clases sin tener que escribir el nombre del paquete (por ejemplo, `Button` en lugar de `mx.controls.Button`).
2. Defina la clase mediante la palabra clave `class`; utilice la palabra clave `extend` para ampliar una clase principal. (Véase [“Definición de la clase y de su superclase” en la página 158.](#))
3. Defina las variables `symbolName`, `symbolOwner` y `className`. (Véase [“Identificación de nombres de clase, símbolo y propietario” en la página 158.](#))
Estas variables sólo son necesarias en los componentes de la versión 2.
4. Defina las variables de miembros. (Véase [“Definición de variables” en la página 159.](#))
Pueden utilizarse en métodos de captador/definidor.
5. Defina una función constructora. (Véase [“Función constructora” en la página 176.](#))
6. Defina un método `init()`. (Véase [“Definición del método `init\(\)`” en la página 174.](#))
Se llama a este método cuando se crea la clase si ésta amplía `UIComponent`. Si la clase amplía `MovieClip`, llame a este método desde la función constructora.
7. Defina un método `createChildren()`. (Véase [“Definición del método `createChildren\(\)`” en la página 174.](#))
Se llama a este método cuando se crea la clase si ésta amplía `UIComponent`. Si la clase amplía `MovieClip`, llame a este método desde la función constructora.
8. Defina un método `size()`. (Véase [“Definición del método `size\(\)`” en la página 177.](#))
Se llama a este método cuando se cambia el tamaño del componente, si la clase amplía `UIComponent`. Además, se llama a este método cuando se cambia el tamaño de la previsualización dinámica del componente durante la edición.
9. Defina un método `draw()`. (Véase [“Invalidación” en la página 178.](#))
Se llama a este método cuando se invalida el componente, si la clase amplía `UIComponent`.

10. Añada una etiqueta de metadatos y una declaración. (Véase [“Adición de metadatos de componente” en la página 161.](#))
Al añadir la etiqueta y la declaración, las propiedades de captador/definidor aparecen en el inspector de propiedades y en el inspector de componentes en Flash.
11. Defina métodos de captador/definidor. (Véase [“Utilización de métodos de captador/definidor para definir parámetros” en la página 160.](#))
12. (Opcional) Cree variables para los elementos y los vínculos de cada aspecto que se utilizan en el componente. (Véase [“Asignación de aspectos” en la página 181.](#))
Esto permite a los usuarios configurar un elemento de aspecto distinto mediante la modificación de un parámetro del componente.

Importación de clases

Puede importar archivos de clase para no tener que escribir nombres de clase completos por todo el código. De esta forma, el código será más conciso y legible. Para importar una clase, utilice la sentencia `import` al principio del archivo de clase, como en el siguiente ejemplo:

```
import mx.core.UIObject;  
import mx.core.ScrollView;  
import mx.core.ext.UIObjectExtensions;  
  
class MyComponent extends UIComponent{
```

También puede utilizar el carácter comodín (*) para importar todas las clases de un paquete determinado. Por ejemplo, la sentencia siguiente importa todas las clases del paquete

```
mx.core:  
import mx.core.*;
```

Si una clase importada no se utiliza en un script, la clase no se incluirá en el código de bytes del archivo SWF resultante. Como resultado, la importación de un paquete entero con un carácter comodín no crea un archivo SWF innecesariamente grande.

Definición de la clase y de su superclase

Un archivo de clase de componente se define como cualquier archivo de clase. Utilice la palabra clave `class` para indicar el nombre de clase. El nombre de clase debe ser también el nombre del archivo de clase. Utilice la palabra clave `extends` para indicar la superclase. Para más información, consulte Capítulo 6, “Escritura de archivos de clases personalizadas” en *Aprendizaje de ActionScript 2.0 en Flash*.

```
class MyComponentName extends UIComponent{  
  
}
```

Identificación de nombres de clase, símbolo y propietario

Para ayudar a Flash a encontrar las clases y los paquetes de ActionScript y para conservar el nombre del componente, debe establecer las variables `symbolName`, `symbolOwner` y `className` en el archivo de clase de ActionScript del componente.

La variable `symbolOwner` es una referencia a un objeto que es un símbolo. Si el componente es su propio `symbolOwner` o si se ha importado `symbolOwner`, no tiene por qué estar completo.

La tabla siguiente describe estas variables:

Variable	Tipo	Descripción
<code>symbolName</code>	String	El nombre de la clase de ActionScript (por ejemplo, <code>ComboBox</code>). Este nombre debe coincidir con el identificador de vinculación del símbolo. Esta variable debe ser estática.
<code>symbolOwner</code>	Object	Nombre de clase completo (por ejemplo, <code>mypackage.MyComponent</code>). No incluya entre comillas el valor <code>symbolOwner</code> , porque su tipo de datos es <code>Object</code> . Este nombre debe coincidir con el de la clase de AS 2.0 del cuadro de diálogo Propiedades de vinculación. Esta variable se utiliza en la llamada interna al método <code>createClassObject()</code> . Esta variable debe ser estática.
<code>className</code>	String	Nombre de la clase del componente. No incluye el nombre del paquete y no tiene una configuración correspondiente en el entorno de desarrollo de Flash. Puede utilizar el valor de esta variable para establecer las propiedades de estilo.

En el siguiente ejemplo se añaden las variables `symbolName`, `symbolOwner` y `className` a la clase `MyButton`:

```
class MyButton extends mx.controls.Button {
    static var symbolName:String = "MyButton";
    static var symbolOwner = myPackage.MyButton;
    var className:String = "MyButton";
}
```

Definición de variables

A continuación se muestra un ejemplo de código del archivo `Button.as` (`mx.controls.Button`). Define una variable, `btnOffset`, que se utilizará en el archivo de clase. También define las variables `__label` y `__labelPlacement`. En las últimas dos variables se añaden dos caracteres de subrayado como prefijo para evitar conflictos de nombres cuando se utilizan en métodos de captador/definidor y como propiedades y parámetros en el componente. Para más información, consulte [“Utilización de métodos de captador/definidor para definir parámetros” en la página 160 en *Aprendizaje de ActionScript 2.0 en Flash*](#).

```
/**
 * Número que se utiliza para desplazar la etiqueta y/o el icono cuando se
 * presiona el botón.
 */
var btnOffset:Number = 0;

/**
 *@private
 * Texto que aparece en la etiqueta si no se especifica ningún valor.
 */
var __label:String = "default value";

/**
 *@private
 * colocación de etiqueta predeterminada
 */
var __labelPlacement:String = "right";
```

Utilización de métodos de captador/definidor para definir parámetros

La forma más sencilla de definir un parámetro de componente es añadir una variable de miembro pública que convierta al parámetro en Inspectable. Para ello, puede utilizar la etiqueta Inspectable en el inspector de componentes o añadir la variable Inspectable del siguiente modo:

```
[Inspectable(defaultValue="strawberry")]  
public var flavorStr:String;
```

Sin embargo, si el código que emplea un componente modifica la propiedad `flavorStr`, normalmente el componente debe realizar una acción para actualizarse como respuesta al cambio de propiedad. Por ejemplo, si se establece `flavorStr` en "cherry", el componente puede volver a dibujarse con una imagen de cereza en lugar de la imagen de fresa predeterminada.

En variables de miembro regulares, el componente no recibe automáticamente notificación del cambio del valor de la variable de miembro.

Los métodos de captador/definidor son una forma directa de detectar cambios en las propiedades de los componentes. En lugar de declarar una variable regular con `var`, declare métodos de captador/definidor, como se muestra a continuación:

```
private var __flavorStr:String = "strawberry";  
  
[Inspectable(defaultValue="strawberry")]  
  
public function get flavorStr():String{  
    return __flavorStr;  
}  
public function set flavorStr(newFlavor:String) {  
    __flavorStr = newFlavor;  
    invalidate();  
}
```

La llamada a `invalidate()` provoca que el componente vuelva a dibujarse con el nuevo sabor. Es la ventaja de utilizar métodos de captador/definidor en la propiedad `flavorStr` en lugar de una variable de miembro regular. Véase [“Definición del método draw\(\)” en la página 177](#).

Para definir métodos de captador/definidor, recuerde estos puntos:

- Añada al nombre de método el prefijo `get` o `set`, seguido de un espacio y del nombre de propiedad.
- La variable que almacena el valor de la propiedad no puede tener el mismo nombre que el captador o el definidor. Normalmente, los nombres de las variables de captador o definidor van precedidos de dos caracteres de subrayado.

Los captadores y los definidores suelen utilizarse junto con etiquetas para definir las propiedades que son visibles en los inspectores de propiedades y de componentes.

Para más información sobre los métodos de captador/definidor, consulte “Métodos `getter` (captador) y `setter` (definidor)” en *Aprendizaje de ActionScript 2.0 en Flash*.

Adición de metadatos de componente

Puede añadir etiquetas de metadatos de componente a los archivos de clase de `ActionScript` externos para indicar al compilador los parámetros del componente, las propiedades de vinculación de datos y los eventos. Las etiquetas de metadatos se utilizan en el entorno de edición de `Flash` para muchos fines diversos.

Las etiquetas de metadatos sólo pueden utilizarse en los archivos de clase de `ActionScript` externos. No se pueden utilizar etiquetas de metadatos en archivos `FLA`.

Los metadatos se asocian con una declaración de clases o con un único campo de datos. Si el valor de un atributo es una cadena, debe ponerlo entre comillas.

Las sentencias de metadatos están vinculadas a la siguiente línea del archivo de `ActionScript`. Cuando defina la propiedad de un componente, añada la etiqueta de metadatos en la línea antes de la declaración de la propiedad. La única excepción es la etiqueta de metadatos `Event`. Cuando defina eventos de los componentes, añada la etiqueta de metadatos fuera de la definición de la clase, de modo que el evento se vincule a toda la clase.

En el ejemplo siguiente, las etiquetas `Inspectable` definen los parámetros `flavorStr`, `colorStr` y `shapeStr`:

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
[Inspectable(defaultValue="blue")]
public var colorStr:String;
[Inspectable(defaultValue="circular")]
public var shapeStr:String;
```

En el inspector de propiedades y en la ficha `Parámetros` del inspector de componentes, `Flash` muestra todos estos parámetros como de tipo `String`.

Etiquetas de metadatos

La tabla siguiente describe las etiquetas de metadatos que puede utilizar en los archivos de clase de ActionScript:

Etiqueta	Descripción
Inspectable	Muestra una propiedad en el inspector de componentes y el inspector de propiedades. Véase “Etiqueta Inspectable” en la página 163 .
InspectableList	Identifica qué subconjunto de propiedades Inspectable deben enumerarse en el inspector de propiedades y el inspector de componentes. Si no añade un atributo InspectableList a la clase del componente, todos los parámetros Inspectable aparecen en el inspector de propiedades. Véase “Etiqueta InspectableList” en la página 165 .
Event	Define un evento de componente. Véase “Etiqueta Event” en la página 165 .
Bindable	Muestra una propiedad en la ficha Vinculaciones del inspector de componentes. Véase “Etiqueta Bindable” en la página 166 .
ChangeEvent	Identifica un evento que provoca la vinculación de datos. Véase “Etiqueta ChangeEvent” en la página 168 .
Collection	Identifica un atributo <code>collection</code> que se muestra en el inspector de componentes. Véase “Etiqueta Collection” en la página 169 .
IconFile	Especifica el nombre de archivo del icono que representa este componente en el panel Componentes. Véase “Etiqueta IconFile” en la página 170 .
ComponentTask	Especifica los nombres de archivo de uno o varios archivos JSFL asociados para realizar tareas en el entorno de edición. Véase “Etiqueta ComponentTask” en la página 170 .

Las secciones siguientes describen con mayor detalle las etiquetas de metadatos de componente.

Etiqueta Inspectable

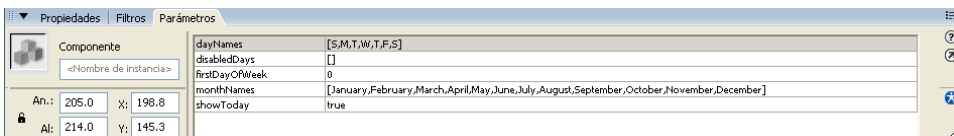
La etiqueta Inspectable se utiliza para especificar los parámetros Inspectable que puede editar el usuario y que aparecen en el inspector de componentes y en el inspector de propiedades. Esto permite mantener las propiedades Inspectable y el código de ActionScript subyacente en la misma ubicación. Para ver las propiedades de los componentes, arrastre una instancia del componente hasta el escenario y seleccione la ficha Parámetros del inspector de componentes.

Los parámetros de colección también son Inspectable. Para más información, consulte [“Etiqueta Collection” en la página 169](#).

La siguiente ilustración muestra la ficha Parámetros del inspector de componentes para el componente DateChooser:



De forma alternativa, puede ver un subconjunto de propiedades del componente en la ficha Parámetros del inspector de propiedades.



Cuando se determinan los parámetros que deben mostrarse en el entorno de edición, Flash utiliza la etiqueta Inspectable. La sintaxis de esta etiqueta es la siguiente:

```
[Inspectable(value_type=value[, attribute=value,...])  
property_declaration name:type;
```

En el siguiente ejemplo se define el parámetro enabled como Inspectable:

```
[Inspectable(defaultValue=true, verbose=1, category="Other")]  
var enabled:Boolean;
```

La etiqueta Inspectable también es compatible con atributos sueltos como el siguiente:

```
[Inspectable("danger", 1, true, maybe)]
```

La sentencia de metadatos debe ir justo antes de la declaración de variables de la propiedad para poder vincularse a dicha propiedad.

En la tabla siguiente se describen los atributos de la etiqueta `Inspectable`:

Atributo	Tipo	Descripción
<code>defaultValue</code>	String o Number	(Opcional) Valor predeterminado para la propiedad <code>Inspectable</code> .
<code>enumeration</code>	String	(Opcional) Especifica una lista separada por comas de valores válidos de la propiedad.
<code>listOffset</code>	Number	(Opcional) Se añade para que sea compatible con los componentes de Flash MX. Se utiliza como el índice predeterminado en un valor <code>List</code> .
<code>name</code>	String	(Opcional) Nombre de visualización de la propiedad. Por ejemplo, <code>Font Width</code> . Si no lo especifica, utilice el nombre de la propiedad, como por ejemplo <code>_fontWidth</code> .
<code>type</code>	String	(Opcional) Especificador del tipo. Si se omite, utilice el tipo de la propiedad. Los valores siguientes son aceptables: <ul style="list-style-type: none">• Array• Boolean• Color• Font Name• List• Number• Object• String
<code>variable</code>	String	(Opcional) Se añade para que sea compatible con los componentes de Flash MX. Especifica la variable con la que está vinculado este parámetro.
<code>verbose</code>	Number	(Opcional) Una propiedad <code>Inspectable</code> que tiene el atributo <code>verbose</code> definido en 1 no aparece en el inspector de propiedades pero sí en el inspector de componentes. Normalmente se utiliza en las propiedades que no se modifican con frecuencia.

Ninguno de estos atributos es necesario; puede utilizar `Inspectable` como etiqueta de metadatos.

Todas las propiedades de la superclase que se marcan como `Inspectable` lo son automáticamente en la clase actual. Utilice la etiqueta `InspectableList` si desea ocultar algunas de estas propiedades para la clase actual.

Etiqueta InspectableList

Utilice la etiqueta `InspectableList` para especificar el subconjunto de propiedades `Inspectable` que debe aparecer en el inspector de propiedades. Utilice `InspectableList` junto con `Inspectable`, de modo que pueda ocultar los atributos heredados para los componentes que son subclases. Si no añade una etiqueta `InspectableList` a la clase del componente, todos los parámetros `Inspectable`, incluidos los de las clases principales del componente, aparecen en el inspector de propiedades.

La sintaxis de `InspectableList` es la siguiente:

```
[InspectableList("attribute1"[,...])]  
// definición de clase
```

La etiqueta `InspectableList` debe ir justo antes de la definición de la clase, ya que se aplica a toda la clase.

En el ejemplo siguiente se muestran las propiedades `flavorStr` y `colorStr` en el inspector de propiedades, pero se excluyen otras propiedades `Inspectable` de la clase `Parent`:

```
[InspectableList("flavorStr", "colorStr")]  
class BlackDot extends DotParent {  
    [Inspectable(defaultValue="strawberry")]  
    public var flavorStr:String;  
    [Inspectable(defaultValue="blue")]  
    public var colorStr:String;  
    ...  
}
```

Etiqueta Event

Utilice la etiqueta `Event` para definir los eventos que emite el componente.

Esta etiqueta tiene la siguiente sintaxis:

```
[Event("event_name")]
```

Por ejemplo, el código siguiente define un evento `click`:

```
[Event("click")]
```

Añada las sentencias `Event` fuera de la definición de la clase en el archivo de `ActionScript`, de modo que se vinculen a la clase y no a un miembro concreto de dicha clase.

En el ejemplo siguiente se muestran los metadatos `Event` para la clase `UIObject`, que gestiona los eventos `resize`, `move` y `draw`:

```
...  
import mx.events.UIEvent;  
[Event("resize")]  
[Event("move")]  
[Event("draw")]  
class mx.core.UIObject extends MovieClip {
```

```
} ...  
}
```

Para difundir una instancia determinada, llame al método `dispatchEvent()`. Véase [“Utilización del método `dispatchEvent\(\)`” en la página 179](#).

Etiqueta Bindable

La vinculación de datos conecta los componentes entre sí. La vinculación de datos visual se consigue a través de la ficha Vinculaciones del inspector de componentes. Desde esta ficha puede añadir, visualizar y eliminar vinculaciones para un componente.

Aunque la vinculación de datos funciona con cualquier componente, su objetivo principal es conectar los componentes de interfaz de usuario a los orígenes de datos externos, como los servicios Web o los documentos XML. Estos orígenes de datos están disponibles como componentes con propiedades, que se pueden vincular con las propiedades de otros componentes.

Utilice la etiqueta `Bindable` antes de una propiedad en una clase de `ActionScript` para que la propiedad aparezca en la ficha Vinculaciones del inspector de componentes. Puede declarar una propiedad mediante `var` o métodos de captador/definidor. Si una propiedad tiene métodos de captador y de definidor, sólo es necesario aplicar la etiqueta `Bindable` a uno de ellos.

La etiqueta `Bindable` tiene la siguiente sintaxis:

```
[Bindable "readonly"|"writeonly",type="datatype"]
```

Ambos atributos son opcionales.

En el ejemplo siguiente se define la variable `flavorStr` como una propiedad a la que se puede acceder en la ficha Vinculaciones del inspector de componentes:

```
[Bindable]  
public var flavorStr:String = "strawberry";
```

La etiqueta Bindable utiliza tres opciones que especifican el tipo de acceso a la propiedad y el tipo de datos de dicha propiedad. La tabla siguiente describe estas opciones:

Opción	Descripción
<code>readonly</code>	Indica que cuando se crean vinculaciones en el inspector de componentes, sólo pueden crearse vinculaciones que utilicen esta propiedad como origen. Sin embargo, esta restricción no se aplica si se utiliza código ActionScript para crear vinculaciones. [Bindable("readonly")]
<code>writable</code>	Indica que cuando se crean vinculaciones en el inspector de componentes, esta propiedad sólo puede utilizarse como destino de una vinculación. Sin embargo, esta restricción no se aplica si se utiliza código ActionScript para crear vinculaciones. [Bindable("writable")]
<code>type="datatype"</code>	Indica el tipo que utiliza la vinculación de datos para la propiedad. El resto de usuarios de Flash utilizan el tipo declarado. Si no especifica esta opción, la vinculación de datos utiliza el tipo de datos de la propiedad como se declara en el código ActionScript. En el siguiente ejemplo, la vinculación de datos tratará <code>x</code> como de tipo <code>DataProvider</code> , aunque sea realmente de tipo <code>Object</code> : [Bindable(type="DataProvider")] <code>var x: Object;</code>

Todas las propiedades de todos los componentes pueden participar en la vinculación de datos. La etiqueta Bindable simplemente controla cuáles de esas propiedades están disponibles para la vinculación en el inspector de componentes. Si una propiedad no va precedida de la etiqueta Bindable, puede seguir utilizándola para la vinculación de datos, pero debe crear las vinculaciones mediante código ActionScript.

La etiqueta Bindable es obligatoria cuando se utiliza la etiqueta ChangeEvent.

Para más información sobre la creación de vinculación de datos en el entorno de edición de Flash, consulte “Vinculación de datos (sólo para Flash Professional)” en *Utilización de Flash*.

Etiqueta ChangeEvent

La etiqueta `ChangeEvent` indica a la vinculación de datos que el componente generará un evento cada vez que cambie el valor de una propiedad específica. Como respuesta al evento, la vinculación de datos ejecuta cualquier vinculación que tenga dicha propiedad como origen. El componente sólo genera el evento si se escribe el código `ActionScript` adecuado en el componente. El evento debería incluirse en la lista de metadatos `Event` declarados por la clase.

Puede declarar una propiedad mediante `var` o métodos de captador/definidor. Si una propiedad tiene métodos de captador y de definidor, sólo es necesario aplicar la etiqueta `Bindable` a uno de ellos.

La etiqueta `ChangeEvent` tiene la siguiente sintaxis:

```
[Bindable]
[ChangeEvent("event")]
property_declaration or getter/setter function
```

En el ejemplo siguiente, el componente genera el evento `change` cuando el valor de la propiedad vinculable `flavorStr` se modifica:

```
[Bindable]
[ChangeEvent("change")]
public var flavorStr:String;
```

Cuando se produce el evento especificado en los metadatos, Flash notifica a las vinculaciones que ha cambiado la propiedad.

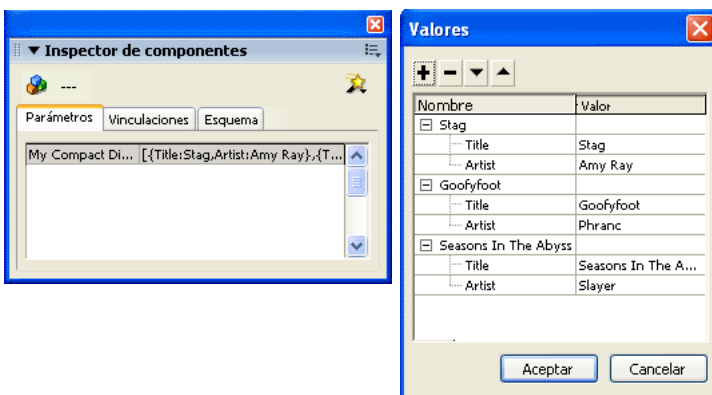
Puede registrar varios eventos en la etiqueta, como se muestra en el ejemplo siguiente:

```
[ChangeEvent("change1", "change2", "change3")]
```

Cualquiera de estos eventos indica un cambio en la propiedad. No es necesario que tengan lugar todos para indicar un cambio.

Etiqueta Collection

Utilice la etiqueta Collection para describir una matriz de objetos que puedan modificarse como una colección de elementos en el cuadro de diálogo Valores durante la edición. El tipo de los objetos se identifica mediante el atributo `collectionItem`. Una propiedad de colección contiene una serie de elementos de colección que se definen en una clase independiente. Esta clase es `mx.utils.CollectionImpl` o una subclase de ella. Para acceder a los objetos individuales se utilizan los métodos de la clase identificada mediante el atributo `collectionClass`.



Propiedad de colección en el inspector de componentes y cuadro de diálogo Valores que aparece al hacer clic en el icono de lupa.

La sintaxis de la etiqueta Collection es la siguiente:

```
[Collection (name="name", variable="varname",  
    collectionClass="mx.utils.CollectionImpl",  
    collectionItem="coll-item-classname", identifier="string")]  
public var varname:mx.utils.Collection;
```

En la tabla siguiente se describen los atributos de la etiqueta Collection:

Atributo	Tipo	Descripción
name	String	(Obligatorio) Nombre de la colección tal y como aparece en el inspector de componentes.
variable	String	(Obligatorio) Variable de ActionScript que señala el objeto Collection subyacente (por ejemplo, un parámetro Collection puede denominarse Columns, pero el atributo variable subyacente puede ser __columns).

Atributo	Tipo	Descripción
<code>collectionClass</code>	String	(Obligatorio) Especifica el tipo de clase del que se va a crear una instancia en la propiedad de colección. Suele ser <code>mx.utils.CollectionImpl</code> , pero también puede ser una clase que amplíe <code>mx.utils.CollectionImpl</code> .
<code>collectionItem</code>	String	(Obligatorio) Especifica la clase de los elementos de colección que se almacenarán en la colección. Esta clase incluye sus propias propiedades <code>Inspectable</code> , que se muestran mediante metadatos.
<code>identifier</code>	String	(Obligatorio) Especifica el nombre de una propiedad <code>Inspectable</code> de la clase del elemento de colección que utiliza Flash MX como identificador predeterminado cuando el usuario añade un nuevo elemento de colección a través del cuadro de diálogo Valores. Cada vez que un usuario crea un nuevo elemento de colección, Flash MX establece el nombre de elemento <i>identifier</i> más un índice único (por ejemplo, si <code>identifier=name</code> , el cuadro de diálogo Valores muestra <code>name0</code> , <code>name1</code> , <code>name2</code> , etc.).

Para más información, consulte [“Propiedades de colección” en la página 203](#).

Etiqueta `IconFile`

Puede añadir un icono que represente el componente en el panel Componentes del entorno de edición de Flash. Para más información, consulte [“Adición de un icono” en la página 199](#).

Etiqueta `ComponentTask`

Puede especificar uno o varios archivos JavaScript de Flash (JSFL) para realizar tareas para el componente desde el entorno de edición de Flash. Utilice la etiqueta `ComponentTask` para definir esta asociación entre el componente y su archivo JSFL y para asociar cualquier archivo adicional necesario para un archivo JSFL. Los archivos JSFL interactúan con la API de JavaScript en el entorno de edición de Macromedia Flash.

NOTA

Los archivos de tareas JSFL y los archivos de dependencia necesarios declarados con la etiqueta `ComponentTask` deben residir en la misma carpeta que el archivo FLA del componente cuando se exporta el componente como un archivo SWC.

La etiqueta `ComponentTask` tiene la siguiente sintaxis:

```
[ComponentTask [taskName,taskFile [,otherFile[,...]]]
```

Los atributos `taskName` y `taskFile` son necesarios. El atributo `otherFile` es opcional.

En el siguiente ejemplo se asocia `SetUp.jsfl` y `AddNewSymbol.jsfl` con la clase de componente denominada `myComponent`. `AddNewSymbol.jsfl` requiere un archivo `testXML.xml` y se especifica en el atributo `otherFile`.

```
[ComponentTask("Do Some Setup","SetUp.jsfl")]
[ComponentTask("Add a new Symbol","AddNewSymbol.jsfl","testXML.xml")]
class myComponent{
    //...
}
```

En la tabla siguiente se describen los atributos de la etiqueta `ComponentTask`:

Atributo	Tipo	Descripción
<code>taskName</code>	String	(Obligatorio) Nombre de la etiqueta en forma de cadena. Este nombre se muestra en el menú emergente Tareas que aparece en la ficha Esquema del inspector de componentes.
<code>taskFile</code>	String	(Obligatorio) Nombre del archivo JSFL que implementa las tareas en el entorno de edición. El archivo debe residir en la misma carpeta que el archivo FLA del componente cuando exporte el componente como un archivo SWC.
<code>otherFile</code>	String	(Opcional) Uno o varios nombres de archivos necesarios por el archivo JSFL como, por ejemplo, un archivo XML. El archivo (o archivos) debe residir en la misma carpeta que el archivo FLA del componente cuando exporte el componente como un archivo SWC.

Definición de parámetros de componentes

Cuando crea un componente, puede añadir parámetros que definan su aspecto y su comportamiento. Los parámetros que se utilizan con más frecuencia aparecen como parámetros de edición en el inspector de componentes y en el inspector de propiedades. También puede establecer todos los parámetros `Inspectable` y `Collection` mediante código `ActionScript`. Para definir estas propiedades en el archivo de clase de componente, utilice la etiqueta `Inspectable` (véase [“Etiqueta Inspectable” en la página 163](#)).

El ejemplo siguiente establece varios parámetros de componentes en el archivo de clase `JellyBean` y los muestra con la etiqueta `Inspectable` en el inspector de componentes:

```
class JellyBean{
    // un parámetro de cadena
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;

    // un parámetro de lista de cadenas
```

```

[Inspectable(enumeration="sour,sweet,juicy,rotten",defaultValue="sweet")
]
public var flavorType:String;

// un parámetro de matriz
[Inspectable(name="Flavors", defaultValue="strawberry,grape,orange",
verbose=1, category="Fruits")]
var flavorList:Array;

// un parámetro de objeto
[Inspectable(defaultValue="belly:flop,jelly:drop")]
public var jellyObject:Object;

// un parámetro de color
[Inspectable(defaultValue="#ffffff")]
public var jellyColor:Color;

// un definidor
[Inspectable(defaultValue="default text")]
function set text(t:String)

}

```

Puede utilizar cualquiera de los siguientes tipos de parámetros:

- Array
- Object
- List
- String
- Number
- Boolean
- Font Name
- Color

NOTA

La clase JellyBean es un ejemplo teórico. Para ver un ejemplo real, consulte el archivo de clase Button.as que se instala con Flash en el directorio *idioma/First Run/Classes/mx/controls*.

Funciones básicas

Debe definir cinco funciones en el archivo de clase del componente: `init()`, `createChildren()`, la función constructora, `draw()` y `size()`. Cuando un componente amplía `UIComponent`, estas cinco funciones del archivo de clase se llaman en el siguiente orden:

- `init()`

Cualquier tarea de inicialización se produce durante la llamada a la función `init()`. Por ejemplo, en este momento pueden establecerse variables de miembro de instancia y puede ocultarse el cuadro de delimitación del componente.

Después de llamar a `init()`, se establecen automáticamente las propiedades `width` y `height`. Véase [“Definición del método `init\(\)`” en la página 174](#).
- `createChildren()`

Se llama durante la reproducción de un fotograma en la línea de tiempo. Durante este tiempo, el usuario del componente puede llamar a los métodos y las propiedades para configurar el componente. Cualquier subobjeto que deba crear el componente se crea con la función `createChildren()`. Véase [“Definición del método `createChildren\(\)`” en la página 174](#).
- Función constructora

Se llama para crear una instancia del componente. La función constructora del componente suele dejarse vacía para evitar conflictos de inicialización. Véase [“Función constructora” en la página 176](#).
- `draw()`

Cualquier elemento visual del componente que se cree o modifique mediante programación debería ocurrir en la función `draw`. Véase [“Definición del método `draw\(\)`” en la página 177](#).
- `size()`

Esta función se llama cuando se cambia el tamaño de un componente en tiempo de ejecución y se pasan propiedades `width` y `height` actualizadas del componente. Con la función `size()`, es posible cambiar el tamaño de los subobjetos del componente o moverlos con respecto a las propiedades `width` y `height` actualizadas del componente. Véase [“Definición del método `size\(\)`” en la página 177](#).

Estas funciones básicas de componentes se describen detalladamente en las siguientes secciones.

Definición del método `init()`

Flash llama al método `init()` cuando se crea la clase. Este método sólo se llama una vez cuando se crea una instancia de un componente.

Debería utilizar el método `init()` para realizar lo siguiente:

- Llame a `super.init()`.
Esto es obligatorio.
- Convierta `boundingBox_mc` en invisible.

```
boundingBox_mc.width = 0;  
boundingBox_mc.height = 0;  
boundingBox_mc.visible = false;
```
- Cree variables de miembro de instancia.

Los parámetros `width`, `height` y `clip` sólo se configuran correctamente después de llamar a este método.

El método `init()` se llama desde el constructor de `UIObject`, de forma que el flujo de control asciende por la cadena de constructores hasta alcanzar a `UIObject`. El constructor de `UIObject` llama al método `init()` que se define en la subclase de menor nivel. Cada implementación de `init()` debería llamar a `super.init()` para que su clase base pueda finalizar la inicialización. Si implementa un método `init()` y no llama a `super.init()`, no se llama al método `init()` en ninguna de las clases base, de modo que es posible que nunca pueda utilizarse.

Definición del método `createChildren()`

Los componentes implementan el método `createChildren()` para crear subobjetos (por ejemplo, otros componentes) en el componente. En lugar de llamar al constructor del subobjeto en el método `createChildren()`, llame a `createClassObject()` o `createObject()` para crear una instancia de un subobjeto del componente.

Es aconsejable llamar a `size()` en el método `createChildren()` para asegurarse de que todos los elementos secundarios se establecen inicialmente con el tamaño correcto. Llame además a `invalidate()` en el método `createChildren()` para actualizar la pantalla. Para más información, consulte [“Invalidación” en la página 178](#).

El método `createClassObject()` utiliza la sintaxis siguiente:

```
createClassObject(className, instanceName, depth, initObject)
```

La tabla siguiente describe los parámetros:

Parámetro	Tipo	Descripción
<i>className</i>	Object	Nombre de la clase.
<i>instanceName</i>	String	Nombre de la instancia.
<i>depth</i>	Number	Profundidad de la instancia.
<i>initObject</i>	Object	Objeto que contiene propiedades de inicialización.

Para llamar a `createClassObject()`, debe saber cuáles son sus elementos secundarios porque es necesario especificar el nombre y el tipo del objeto además de los parámetros de inicialización en la llamada a `createClassObject()`.

En el ejemplo siguiente se llama a `createClassObject()` para crear un nuevo objeto `Button` que se utilizará en un componente:

```
up_mc.createClassObject(mx.controls.Button, "submit_btn", 1);
```

Para establecer propiedades en la llamada a `createClassObject()`, añádalas como parte del parámetro *initObject*. En el ejemplo siguiente se define el valor de la propiedad `label`:

```
form.createClassObject(mx.controls.CheckBox, "cb", 0, {label:"Check  
this"});
```

En el ejemplo siguiente se crean los componentes `TextInput` y `SimpleButton`:

```
function createChildren():Void {  
    if (text_mc == undefined)  
        createClassObject(TextInput, "text_mc", 0, { preferredWidth: 80,  
            editable:false });  
        text_mc.addEventListener("change", this);  
        text_mc.addEventListener("focusOut", this);  
  
    if (mode_mc == undefined)  
        createClassObject(SimpleButton, "mode_mc", 1, { falseUpSkin:  
            modeUpSkinName, falseOverSkin: modeOverSkinName, falseDownSkin:  
            modeDownSkinName });  
        mode_mc.addEventListener("click", this);  
        size()  
        invalidate()  
}
```

Función constructora

Puede reconocer una función constructora porque tiene el mismo nombre que la clase del componente. Por ejemplo, el código siguiente muestra la función constructora del componente `ScrollBar`:

```
function ScrollBar() {  
}
```

En este caso, cuando se crea una nueva instancia de barra de desplazamiento, se llama al constructor `ScrollBar()`.

Normalmente, los constructores de componentes deberían estar vacíos. A veces, al configurar las propiedades en los constructores se sobrescriben los valores predeterminados, en función del orden de las llamadas de inicialización.

Si el componente amplía `UIComponent` o `UIObject`, Flash llama automáticamente a los métodos `init()`, `createChildren()` y `size()` y puede dejarse vacía la función constructora, como se muestra a continuación:

```
class MyComponent extends UIComponent{  
    ...  
    // es la función constructora  
    function MyComponent(){  
    }  
}
```

Todos los componentes de la versión 2 deberían definir una función `init()` que se llame después de llamar al constructor. Debería incluir el código de inicialización en la función `init()` del componente. Para más información, consulte la sección siguiente.

Si su componente amplía `MovieClip`, quizás desee llamar a un método `init()`, a un método `createChildren()` y a un método que disponga el componente desde la función constructora, tal y como se muestra en el siguiente ejemplo de código:

```
class MyComponent extends MovieClip{  
    ...  
    function MyComponent(){  
        init()  
    }  
  
    function init():Void{  
        createChildren();  
        layout();  
    }  
    ...  
}
```

Para más información sobre constructores, consulte “Escritura de la función constructora” en *Aprendizaje de ActionScript 2.0 en Flash*.

Definición del método draw()

Puede escribir código en el método `draw()` para crear o modificar elementos visuales de un componente. Dicho de otro modo, en el método `draw()`, un componente se dibuja a sí mismo para que sus variables de estado coincidan. Es posible que se haya llamado a varias propiedades o métodos desde la última llamada al método `draw()` y debería tratar de tenerlas en cuenta en el cuerpo de `draw()`.

Sin embargo, no debería llamar directamente al método `draw()`. En su lugar, llame al método `invalidate()` para que las llamadas a `draw()` puedan ponerse en cola y gestionarse en un lote. Este enfoque aumenta la eficacia y centraliza el código. Para más información, consulte [“Invalidación” en la página 178](#).

En el método `draw()` puede utilizar llamadas a la interfaz API de dibujo de Flash para dibujar bordes, reglas y otros elementos gráficos. También puede establecer valores de propiedades y llamar a métodos. O puede llamar al método `clear()`, que elimina los objetos visibles.

En el ejemplo siguiente del componente `Dial` (véase [“Creación del primer componente” en la página 136](#)), el método `draw()` establece la rotación del indicador en la propiedad `value`:

```
function draw():Void {
    super.draw();
    dial.needle._rotation = value;
}
```

Definición del método size()

Cuando se cambia el tamaño de un componente en tiempo de ejecución mediante el método `componentInstance.setSize()`, se invoca la función `size()` y se pasan las propiedades `width` y `height`. Puede utilizar el método `size()` en el archivo de clase del componente para disponer el contenido del componente.

Como mínimo, el método `size()` debe llamar al método `size()` de la superclase (`super.size()`).

En el siguiente ejemplo del componente `Dial` (véase [“Creación del primer componente” en la página 136](#)), el método `size()` utiliza los parámetros `width` y `height` para cambiar el tamaño del clip de película de `dial`:

```
function size():Void {
    super.size();
    dial._width = width;
    dial._height = height;
    invalidate();
}
```

Llame al método `invalidate()` en el método `size()` para etiquetar el componente para volver a dibujarlo en lugar de llamar directamente al método `draw()`. Para más información, consulte la sección siguiente.

Invalidación

En la mayoría de los casos, Macromedia recomienda que el componente no se actualice a sí mismo inmediatamente sino que tenga una copia del nuevo valor de la propiedad, establezca un indicador para marcar lo que ha cambiado y llame al método `invalidate()`. (Este método indica que sólo ha cambiado el aspecto visual del objeto y que no ha cambiado el tamaño ni la posición de los subobjetos. Este método llama al método `draw()`.)

Debe llamar a un método de invalidación al menos una vez durante la creación de instancias del componente. Lo más habitual es hacerlo en los métodos `createChildren()` o `layoutChildren()`.

Distribución de eventos

Si desea que el componente difunda eventos distintos a los que hereda de su clase principal, debe llamar al método `dispatchEvent()` en el archivo de clase del componente.

El método `dispatchEvent()` se define en la clase `mx.events.EventDispatcher` y todos los componentes que amplían `UIObject` lo heredan. (Consulte “Clase `EventDispatcher`” en *Referencia del lenguaje de componentes*.)

También debería añadir una etiqueta de metadatos `Event` en la parte superior del archivo de clase para cada nuevo evento. Para más información, consulte “[Etiqueta Event](#)” en [la página 165](#).

NOTA

Para más información sobre la gestión de eventos de componente en una aplicación Flash, consulte el [Capítulo 4](#), “[Gestión de eventos de componentes](#)”, en [la página 69](#).

Utilización del método `dispatchEvent()`

En el cuerpo del archivo de clase `ActionScript` del componente, difunde eventos mediante el método `dispatchEvent()`. El método `dispatchEvent()` utiliza la sintaxis siguiente:

```
dispatchEvent(eventObj)
```

El parámetro `eventObj` es un objeto de `ActionScript` que describe el evento (véase el ejemplo que se muestra más adelante en esta sección).

Debe declarar el método `dispatchEvent()` en el código antes de llamarlo, del siguiente modo:

```
private var dispatchEvent:Function;
```

También debe crear un objeto de evento para pasarlo a `dispatchEvent()`. El objeto de evento contiene información sobre el evento que el detector puede utilizar para gestionar el evento.

Puede crear de forma explícita un objeto de evento antes de distribuir el evento, como se muestra en el ejemplo siguiente:

```
var eventObj = new Object();
eventObj.type = "myEvent";
eventObj.target = this;
dispatchEvent(eventObj);
```

También puede utilizar una sintaxis de método abreviado que establezca el valor de las propiedades `type` y `target` y distribuya el evento en una sola línea:

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

En el ejemplo anterior, no es obligatorio definir la propiedad `target`, ya que es implícita.

La descripción de cada evento en la documentación de Flash 8 muestra una lista con las propiedades de eventos opcionales y las obligatorias. Por ejemplo, el evento `ScrollBar.scroll` utiliza una propiedad `detail` además de las propiedades `type` y `target`. Para más información, consulte las descripciones de los eventos en *Referencia del lenguaje de componentes*.

Eventos frecuentes

En la tabla siguiente se enumeran los eventos comunes difundidos por diversas clases. Cada componente debe difundir estos eventos si tienen algún significado para ese componente. Esta lista no incluye los eventos de todos los componentes, sino solamente los que otros componentes pueden utilizar. Aunque algunos eventos no especifican ningún parámetro, todos los eventos cuentan con un parámetro implícito: una referencia al objeto que está difundiendo el evento.

Evento	Utilización
<code>click</code>	Utilizado por el componente <code>Button</code> o siempre que un clic del ratón no tenga otro significado.
<code>change</code>	Utilizado por <code>List</code> , <code>ComboBox</code> y otros componentes de entrada de texto.
<code>scroll</code>	Utilizado por <code>ScrollBar</code> y por otros controles de desplazamiento (puntos de desplazamiento en un menú emergente de desplazamiento).

Asimismo, debido a la herencia de las clases base, todos los componentes difunden los eventos siguientes:

Evento <code>UIComponent</code>	Descripción
<code>load</code>	El componente está creando o cargando sus subobjetos.
<code>unload</code>	El componente está descargando sus subobjetos.
<code>focusIn</code>	Ahora el componente tiene la selección de entrada. Algunos componentes equivalentes a HTML (<code>ListBox</code> , <code>ComboBox</code> , <code>Button</code> , <code>Text</code>) también pueden difundir <code>focus</code> , pero todos difunden <code>DOMFocusIn</code> .
<code>focusOut</code>	El componente ha perdido la selección de entrada.
<code>move</code>	El componente se ha desplazado a otra ubicación.
<code>resize</code>	Se ha cambiado el tamaño del componente.

La tabla siguiente describe eventos clave comunes:

Eventos clave	Descripción
<code>keyDown</code>	Se presiona una tecla. La propiedad <code>code</code> contiene el código clave y la propiedad <code>ascii</code> contiene el código ASCII de la tecla que se ha presionado. No consulte el objeto <code>Key</code> de bajo nivel, ya que éste tal vez no haya generado el evento.
<code>keyUp</code>	Se suelta una tecla.

Asignación de aspectos

Un componente de interfaz de usuario está formado en su totalidad por clips de película adjuntos. Esto significa que todos los activos de un componente de interfaz de usuario pueden ser externos al clip de película del componente de interfaz de usuario, de modo que otros componentes puedan utilizarlos. Por ejemplo, si el componente necesita funcionalidad de casilla de verificación, puede reutilizar los activos del componente CheckBox existente.

El componente CheckBox utiliza un clip de película independiente para representar todos sus estados (FalseUp, FalseDown, Disabled, Selected, etc.). Sin embargo, puede asociar los clips de película personalizados, llamados *aspectos*, con estos estados. En tiempo de ejecución, los clips de película antiguos y nuevos se exportan al archivo SWF. Los estados antiguos pasan a ser invisibles para dejar paso a los nuevos clips de película. Esta capacidad para cambiar los aspectos durante la edición y en tiempo de ejecución se denomina *aplicación de aspectos*.

Para aplicar aspectos a los componentes, cree una variable para cada elemento de aspecto (símbolo de clip de película) que se utiliza en el componente y establézcala en el identificador de vinculación del símbolo. Esto permite al desarrollador establecer un elemento de aspecto distinto simplemente modificando un parámetro del componente, como se muestra a continuación:

```
var falseUpIcon = "mySkin";
```

El ejemplo siguiente muestra las variables de aspecto para los diversos estados del componente CheckBox:

```
var falseUpSkin:String = "";
var falseDownSkin:String = "";
var falseOverSkin:String = "";
var falseDisabledSkin:String = "";
var trueUpSkin:String = "";
var trueDownSkin:String = "";
var trueOverSkin:String = "";
var trueDisabledSkin:String = "";
var falseUpIcon:String = "CheckFalseUp";
var falseDownIcon:String = "CheckFalseDown";
var falseOverIcon:String = "CheckFalseOver";
var falseDisabledIcon:String = "CheckFalseDisabled";
var trueUpIcon:String = "CheckTrueUp";
var trueDownIcon:String = "CheckTrueDown";
var trueOverIcon:String = "CheckTrueOver";
var trueDisabledIcon:String = "CheckTrueDisabled";
```

Estilos

Puede utilizar estilos para registrar todos los gráficos del componente con una clase y permitir que la clase controle el esquema de colores de los gráficos en tiempo de ejecución. No es necesario ningún código especial en las implementaciones del componente para admitir los estilos. Los estilos se implementan en su totalidad en las clases base (UIObject y UIComponent) y los aspectos.

Para añadir un nuevo estilo a un componente, llame a `getStyle("styleName")` en la clase del componente. Si se ha establecido el estilo en una instancia en una hoja de estilos personalizada o en la hoja de estilos global, se recupera el valor. De lo contrario, es posible que sea necesario instalar un valor predeterminado para el estilo en la hoja de estilos global.

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 86.](#)

Registro de aspectos en estilos

En el siguiente ejemplo se crea un componente denominado Shape. Este componente muestra una forma que tiene uno de los dos aspectos posibles: un círculo o un cuadrado. Los aspectos se registran en el estilo `themeColor`.

Para registrar un aspecto en un estilo:

1. Cree un nuevo archivo de ActionScript y copie en él el siguiente código:

```
import mx.core.UIComponent;

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();
    }
}
```

2. Guarde el archivo como Shape.as.
3. Cree un nuevo documento de Flash y guárdelo como Shape.fla en la misma carpeta que Shape.as.
4. Dibuje un círculo en el escenario, selecciónelo y presione F8 para convertirlo en un clip de película.
Asigne al círculo el nombre e identificador de vinculación **circle_skin**.
5. Abra el clip de película `circle_skin` y coloque el siguiente código ActionScript en el fotograma 1 para registrar el símbolo con el nombre de estilo `themeColor`:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");
```
6. Cree un nuevo clip de película para el componente.
Asigne al clip de película el nombre y el identificador de vinculación **Shape**.
7. Cree dos capas. Coloque una función `stop()` en el primer fotograma de la primera capa. Coloque el símbolo `circle_skin` en el segundo fotograma.
Éste es el clip de película del componente. Para más información, consulte [“Creación de un clip de película del componente” en la página 149](#).
8. Abra `StandardComponents.fla` como una biblioteca externa y arrastre el clip de película `UIComponent` al escenario en el segundo fotograma del clip de película `Shape` (con `circle_skin`).
9. Cierre `StandardComponents.fla`.
10. Seleccione el clip de película `Shape` en la biblioteca y elija Definición de componente en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada) e introduzca el nombre de clase de AS 2.0 **Shape**.
11. Pruebe el clip de película con el componente `Shape` en el escenario.
Para cambiar el color de tema, establezca el estilo en la instancia. El código siguiente cambia a rojo el color de un componente `Shape` con el nombre de instancia `shape`:

```
shape.setStyle("themeColor",0xff0000);
```
12. Dibuje un cuadrado en el escenario y conviértalo en un clip de película.
Asígnele el nombre de vinculación **square_skin** y asegúrese de que está activada la casilla de verificación Exportar en primer fotograma.

NOTA

Como el clip de película no se encuentra en el componente, debe seleccionarse la opción Exportar en primer fotograma para que el aspecto esté disponible antes de la inicialización.

- 13.** Abra el clip de película `square_skin` y coloque el siguiente código ActionScript en el fotograma 1 para registrar el símbolo con el nombre de estilo `themeColor`:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");
```

- 14.** Coloque el siguiente código en la instancia del componente `Shape` en el escenario, en la línea de tiempo principal:

```
onClipEvent(initialize){  
    themeShape = "square_skin";  
}
```

- 15.** Pruebe el clip de película con `Shape` en el escenario. Debería mostrarse un cuadrado rojo.

Registro de un nuevo nombre de estilo

Si ha creado un nuevo nombre de estilo y se trata de un estilo de color, añada el nuevo nombre al objeto `colorStyles` del archivo `StyleManager.as` (First

Run\Classes\mx\styles\StyleManager.as). En este ejemplo se añade el estilo `shapeColor`:

```
// inicializar conjunto de estilos de color heredados  
static var colorStyles:Object =  
{  
    barColor: true,  
    trackColor: true,  
    borderColor: true,  
    buttonColor: true,  
    color: true,  
    dateHeaderColor: true,  
    dateRollOverColor: true,  
    disabledColor: true,  
    fillColor: true,  
    highlightColor: true,  
    scrollTrackColor: true,  
    selectedDateColor: true,  
    shadowColor: true,  
    strokeColor: true,  
    symbolBackgroundColor: true,  
    symbolBackgroundDisabledColor: true,  
    symbolBackgroundPressedColor: true,  
    symbolColor: true,  
    symbolDisabledColor: true,  
    themeColor:true,  
    todayIndicatorColor: true,  
    shadowCapColor:true,  
    borderCapColor:true,  
    focusColor:true,  
    shapeColor:true  
};
```


Registre el nuevo nombre de estilo en los aspectos de círculo y cuadrado en el fotograma 1 de cada clip de película de aspecto, como se muestra a continuación:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "shapeColor");
```

Puede cambiarse el color en el nuevo nombre de estilo si se establece el estilo en la instancia, como se muestra a continuación:

```
shape.setStyle("shapeColor",0x00ff00);
```

Incorporación de componentes existentes en el componente

En esta sección, creará un componente LogIn sencillo que incorpore componentes Label, TextInput y Button. Este tutorial muestra cómo incorporar componentes existentes en los componentes nuevos añadiendo sus símbolos de biblioteca de Flash (FLA) sin compilar. El conjunto de archivos del componente, LogIn.fla, LogIn.as y LogIn.swf, se ubica en la carpeta de ejemplos del disco duro:

- En Windows: carpeta C:\Archivos de programa\Macromedia\FIash8\Samples and Tutorials\Samples\Components\Login.
- En Macintosh: carpeta Disco duro/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Login.

El componente LogIn proporciona una interfaz para introducir un nombre y una contraseña. La API de LogIn tiene dos propiedades, `name` y `password`, para establecer y obtener los valores de cadena en los campos TextInput de nombre y contraseña. El componente LogIn también distribuye un evento “click” cuando el usuario hace clic en un botón con la etiqueta LogIn.

- [“Creación del archivo LogIn Flash \(FLA\)” en la página 186](#)
- [“Archivo de clase LogIn” en la página 189](#)
- [“Prueba y exportación del componente LogIn” en la página 193](#)

Creación del archivo LogIn Flash (FLA)

Para empezar, cree un archivo de Flash (FLA) donde se incluirá el símbolo del componente.

Para crear el archivo LogIn FLA:

1. En Flash, seleccione Archivo > Nuevo y cree un nuevo documento.
2. Seleccione Archivo > Guardar como y guarde el archivo como LogIn fla.
3. Seleccione Insertar > Nuevo símbolo. Asígnele el nombre **LogIn** y seleccione el botón de opción de tipo clip de película.
Si la sección Vinculación del cuadro de diálogo Crear un nuevo símbolo no se abre, haga clic en el botón Avanzado para mostrarla.
4. Active la opción Exportar para ActionScript y desactive Exportar en primer fotograma.
5. Introduzca un identificador de vinculación.
El identificador de vinculación predeterminado es LogIn. En el resto de pasos se supone que se utilizan los valores predeterminados.
6. Introduzca **LogIn** en el cuadro de texto Clase de AS 2.0. Este valor es el nombre de la clase del componente.
Si incluye la clase en un paquete, introduzca el nombre completo del paquete. Por ejemplo, mx.controls.CheckBox denota la clase CheckBox en el paquete mx.controls.
7. Haga clic en Aceptar.
Flash se abrirá en el modo de edición de símbolos.
8. Inserte una nueva capa. Denomine a la capa superior **Acciones** y a la capa inferior **Activos**.
9. Seleccione el fotograma 2 de la capa Activos e inserte un fotograma clave (F6).
La estructura del clip de película del componente consta de dos elementos: una capa Acciones y una capa Activos. La capa Acciones tiene un fotograma clave y la capa Activos tiene dos fotogramas clave.
10. Seleccione el fotograma 1 de la capa Acciones y abra el panel Acciones (F9). Introduzca una función global `stop()`.
De esta forma evitará que el clip de película continúe en el fotograma 2.

11. Seleccione Archivo > Importar > Abrir biblioteca externa y elija el archivo StandardComponents.fla de la carpeta Configuration/ComponentFLA.
 - En Windows: \Archivos de programa\Macromedia\Flash 8\idioma\Configuration\ComponentFLA\StandardComponents.fla.
 - En Macintosh: Disco duro/Applications/Macromedia Flash 8/Configuration/ComponentFLA/StandardComponents.fla

NOTA

Para más información sobre las ubicaciones de carpeta, consulte “Carpetas de configuración instaladas con Flash” en Utilización de Flash.

12. Seleccione el fotograma 2 en la capa Activos. Desde la biblioteca StandardComponents.fla, navegue a la carpeta Flash UI Components 2. Arrastre un símbolo de componente Button, Label y TextInput al fotograma 2 de la capa Activos.

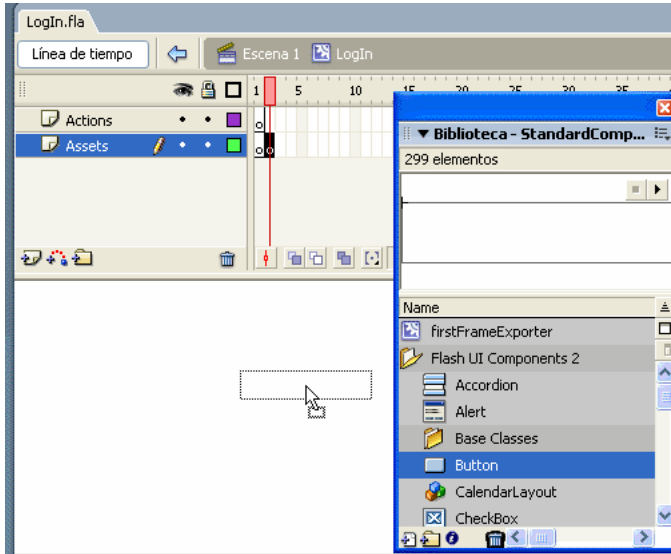
Las dependencias de activos de estos componentes se copian automáticamente a la biblioteca LogIn.fla.

Todos los activos del componente se añaden al fotograma 2 de la capa Activos. Como hay una función global `stop()` en el fotograma 1 de la capa Acciones, los activos del fotograma 2 no se verán mientras se organizan en el escenario.

Se añaden activos al fotograma 2 por dos motivos:

- Para que todos los activos se copien automáticamente a la biblioteca y estén disponibles para crear dinámicamente una instancia y acceder a sus métodos, propiedades y eventos.

- Para asegurarse, mediante la colocación de activos en un fotograma, de que se cargan sin problemas mientras se reproduce la película, evitando tener que establecer la exportación de activos de la biblioteca antes del primer fotograma. Este método evita un pico inicial de transferencia de datos que podría provocar demoras o pausas largas durante la descarga.



Arrastrar un símbolo de componente Button desde la biblioteca en StandardComponents.fla al fotograma 2 de la capa Activos de LogIn.fla

13. Cierre la biblioteca StandardComponents.fla.
14. Seleccione el fotograma 1 de la capa Activos. Arrastre el clip de película BoundingBox de la biblioteca LogIn.fla (carpeta Component Assets) al escenario.
15. Asigne a BoundingBox el nombre de instancia **boundingBox_mc**.

16. Utilice el panel Información para cambiar el tamaño de BoundingBox hasta igualarlo con el del clip de película LogInFinal (340,150) y colocarlo en la posición a 0,0.

La instancia de BoundingBox se utiliza para crear la previsualización dinámica del componente y permite al usuario cambiar el tamaño del mismo durante la edición. Debe establecer el tamaño del cuadro de delimitación para que pueda contener todos los elementos gráficos del componente.

NOTA

Si amplía un componente (incluido cualquier componente de la versión 2), debe mantener los nombres de instancia que ya utilice ese componente, ya que su código hará referencia a dichos nombres de instancia. Por ejemplo, si incluye un componente de la versión 2 que ya utilice el nombre de instancia boundingBox_mc, no cambie el nombre. En sus propios componentes, puede utilizar cualquier nombre de instancia que sea exclusivo y no entre en conflicto con ninguno de los que ya existen en el mismo ámbito.

17. Seleccione el clip de película LogIn en la biblioteca y elija Definición de componente en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
18. En el cuadro de texto Clase de AS 2.0, introduzca **LogIn**.
Este valor es el nombre de la clase de ActionScript. Si la clase está en un paquete, el valor es el paquete completo. Por ejemplo, mx.controls.CheckBox denota la clase CheckBox en el paquete mx.controls.
19. Haga clic en Aceptar.
20. Guarde el archivo.

Archivo de clase LogIn

A continuación se muestra el código de la clase de ActionScript para el componente LogIn. Lea los comentarios del código para obtener una descripción de cada sección. Para más información sobre los elementos de un archivo de clase de componente, consulte [“Información general sobre el archivo de clase de un componente” en la página 156](#).

Para crear este archivo, puede crear un nuevo archivo de ActionScript en Flash o utilizar cualquier otro editor de texto. Guarde el archivo como LogIn.as en la misma carpeta que el archivo LogIn fla.

Puede copiar o escribir el siguiente código de la clase de ActionScript del componente LogIn en su nuevo archivo LogIn.as. Si escribe el código en lugar de copiarlo, podrá familiarizarse con cada elemento del código de componente.

```
/* Importar los paquetes para que se puede hacer referencia a ellos
   directamente desde esta clase. */
import mx.core.UIComponent;
import mx.controls.Label;
import mx.controls.TextInput;
import mx.controls.Button;

// Etiqueta de metadatos Event
[Event("change")]
[Event("click")]
class LogIn extends UIComponent
{
    /* Los componentes deben declarar estas variables de miembros para ser
       componentes adecuados del marco de componentes. */
    static var symbolName:String = "LogIn";
    static var symbolOwner:Object = LogIn;
    var className:String = "LogIn";

    // La representación gráfica del componente.
    private var name_label:MovieClip;
    private var password_label:MovieClip;
    private var name_ti:MovieClip;
    private var password_ti:MovieClip;
    private var login_btn:MovieClip;
    private var boundingBox_mc:MovieClip;
    private var startDepth:Number = 10;

    /* Variables de miembros privadas disponibles públicamente mediante
       captadores/definidores.
       Representan los valores de cadena InputText de nombre y contraseña. */
    private var __name:String;
    private var __password:String;

    /* Constructor:
       Necesario para todas las clases; en componentes de la versión 2
       el constructor debe tener cero argumentos.
       Toda la inicialización se produce en un método init
       necesario después de construir la instancia de la clase. */
    function LogIn() {
    }

    /* Código de inicialización:
       El método init es necesario en componentes de la versión 2. Debe llamar
       también
       al método init() de su clase principal con super.init().
       El método init es necesario en componentes que amplían UIComponent. */
```

```

function init():Void {
    super.init();
    boundingBox_mc._visible = false;
    boundingBox_mc._width = 0;
    boundingBox_mc._height = 0;
}

/* Crear objetos secundarios necesarios al iniciar:
   El método createChildren es necesario en componentes
   que amplían UIComponent. */
public function createChildren():Void {
    name_label = createObject("Label", "name_label", this.startDepth++);
    name_label.text = "Name:";
    name_label._width = 200;
    name_label._x = 20;
    name_label._y = 10;

    name_ti = createObject("TextInput", "name_ti",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:30});
    name_ti.html = false;
    name_ti.text = __name;
    name_ti.tabIndex = 1;
    /* Establecer selección de este campo de entrada de texto.
       Nota: asegurarse de seleccionar Control > Deshabilitar métodos
       abreviados de teclado
       en el depurador de Flash si no está seleccionado; de lo contrario
       es posible que no se establezca la selección durante la prueba. */
    name_ti.setFocus();

    name_label = createObject("Label", "password_label",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:60});
    name_label.text = "Password:";

    password_ti = createObject("TextInput", "password_ti",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:80});
    password_ti.html = false;
    password_ti.text = __password;
    password_ti.password = true;
    password_ti.tabIndex = 2;

    login_btn = createObject("Button", "login_btn",
this.startDepth++,{_width:80,_heigh:22,_x:240,_y:80});
    login_btn.label = "LogIn";
    login_btn.tabIndex = 3;
    login_btn.addEventListener("click", this);

    size();
}

```

```

/* El método draw es necesario en componentes de la versión 2.
   Se invoca tras invalidar el componente
   mediante una llamada a invalidate().
   Es mejor agrupar los cambios en un nuevo dibujo que
   hacerlos individualmente. Este enfoque permite
   centralizar más y mejor el código. */
function draw():Void {
    super.draw();
}

/* El método size se invoca cuando cambia el tamaño del
   componente. Es una oportunidad para cambiar el tamaño de elementos
   secundarios.
   El método size es necesario en componentes que amplían UIComponent. */
function size():Void {
    super.size();
    // Provocar un nuevo dibujo si es necesario.
    invalidate();
}

/* Controlador de eventos:
   Se llama con el botón LogIn al hacer clic en él con el ratón.
   Como se desea que este evento sea accesible fuera del ámbito de
   este componente, el evento click se distribuye mediante dispatchEvent.
*/
public function click(evt){
    // Actualizar las variables de miembros con el contenido del campo de
    entrada.
    __name = name_ti.text;
    __password = password_ti.text;
    // Distribuir un evento click cuando el botón lo active.
    dispatchEvent({type:"click"});
}

/* Es la función de captador/definidor de la propiedad name.
   Los metadatos [Inspectable] hacen que la propiedad aparezca
   en el inspector de propiedades y permite establecer
   un valor predeterminado. Con métodos de captador/definidor puede
   realizar llamadas invalidate
   y hacer que se redibuje el componente cuando se cambia el valor. */
[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue="")]
function set name(val:String){
    __name = val;
    invalidate();
}

function get name():String{
    return(__name);
}

```



```

}

[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue="")]
function set password(val:String){
    __password=val;
    invalidate();
}

function get password():String{
    return(__password);
}
}

```

Prueba y exportación del componente LogIn

Ha creado el archivo de Flash que contiene los elementos gráficos y las clases base, y el archivo de clase que contiene toda la funcionalidad del componente LogIn. Es el momento de probar el componente.

Lo ideal sería ir probando el componente mientras se trabaja, especialmente mientras se escribe el archivo de clase. La forma más rápida de realizar pruebas mientras se trabaja es convertir el componente en un clip compilado y utilizarlo en el archivo FLA del componente.

Cuando haya terminado de crear el componente, expórtelo como un archivo SWC. Para más información, consulte [“Exportación y distribución de un componente” en la página 195](#).

Para probar el componente LogIn:

1. En el archivo LogIn.fla, elija el clip de película LogIn en la biblioteca y seleccione Convertir en clip compilado en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).

Se añadirá a la biblioteca un clip compilado denominado LogIn SWF. Está compilando el clip de película únicamente para probarlo. De lo contrario, seguiría las instrucciones que se indican más adelante en esta sección para exportar el clip de película LogIn.

NOTA

Si ya ha creado un clip compilado (por ejemplo, si es la segunda o tercera vez que realiza pruebas), aparecerá un cuadro de diálogo Solucionar conflicto de biblioteca. Seleccione Reemplazar elementos existentes para añadir la nueva versión del documento.

2. Arrastre el archivo `LogIn SWF` al escenario, en el fotograma 1 de la línea de tiempo principal (asegúrese de que se encuentra en la escena 1 de la línea de tiempo principal y no en la línea de tiempo del clip de película).

Puede cambiar la propiedad `name` y `password` en la ficha `Parámetros` o en el inspector de componentes. Resulta útil si desea que aparezca un texto predeterminado como “Escriba aquí su nombre” antes de que el usuario escriba nada. Cuando haya establecido la propiedad `name` y/o `password`, el texto predeterminado de los subcomponentes `InputText` de nombre y contraseña cambiarán de forma correspondiente en tiempo de ejecución.

Para probar la propiedad `value` en tiempo de ejecución, asigne a la instancia de `LogIn` en el escenario el nombre `myLogin` y añada el siguiente código al fotograma 1 de la línea de tiempo principal:

```
// Crea un campo de texto donde pueden verse los valores de LogIn.
createTextField("myLoginValues",10,10,10,340,40)
myLoginValues.border = true;
// Controlador de eventos del evento click distribuido por la instancia
del componente LogIn.
function click(evt){
/* Aquí tendría lugar la autenticación.
Por ejemplo, el nombre y la contraseña se pasarían a un servicio Web
que los autentique y devuelva un ID de sesión y/o permisos atribuidos
al usuario. */
myLoginValues.text = "Processing...\r";
myLoginValues.text += "Name: " + myLogin.name + " Password: " +
myLogin.password;
}

myLogin.addEventListener("click",this);
```

3. Seleccione `Control > Probar película` para probar el componente en Flash Player.

NOTA

Como está probando este componente en el documento original, es posible que aparezca un mensaje de advertencia que le indique que hay un mismo identificador de vinculación para dos símbolos. El componente funcionará con normalidad. En la práctica, deberá utilizar el nuevo componente con otro documento, en cuyo caso el identificador de vinculación será único.

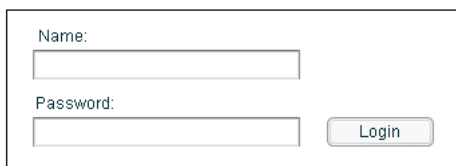
Para exportar el componente LogIn:

1. En el archivo LogIn fla, elija el clip de película LogIn en la biblioteca y seleccione Definición de componente en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
2. Active la opción Mostrar en el panel de componentes.
3. Haga clic en Aceptar.
4. En el archivo LogIn fla, elija el clip de película LogIn en la biblioteca y seleccione Exportar archivo SWC en el menú contextual Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
5. Seleccione la ubicación donde se guardará el archivo SWC.

Si se guarda en la carpeta Components de la carpeta de configuración de usuario, puede volver a cargar el panel Componentes sin reiniciar Flash y el componente aparecerá en el panel Componentes.

NOTA

Para más información sobre las ubicaciones de carpeta, consulte “Carpetas de configuración instaladas con Flash” en *Primeros pasos con Flash*.



The image shows a rectangular form with a thin border. It contains two text input fields. The first field is labeled "Name:" and the second is labeled "Password:". To the right of the "Password:" field is a button labeled "Login".

El componente LogIn completo

Exportación y distribución de un componente

Flash exporta componentes como paquetes de componentes (archivos SWC). Los componentes pueden distribuirse como archivos SWC o archivos FLA. (Consulte el artículo de Macromedia DevNet en www.macromedia.com/support/flash/applications/creating_comps/creating_comps12.html para obtener información sobre la distribución de un componente como un archivo FLA.)

La mejor forma de distribuir un componente es exportarlo como un archivo SWC porque los archivos SWC contienen todo el código ActionScript, archivos SWF y otros archivos opcionales necesarios para utilizar el componente. Los archivos SWC también son útiles si se trabaja simultáneamente en un componente y en la aplicación que utiliza el componente.

Los archivos SWC pueden emplearse para distribuir componentes que se utilizarán en Macromedia Flash 8, Macromedia Dreamweaver MX 2004 y Macromedia Director MX 2004.

Si desarrolla un componente para su propio uso o para que lo utilicen otras personas, es importante probar el archivo SWC como una parte continua del desarrollo de componentes. Por ejemplo, pueden surgir problemas en un archivo SWC de un componente que no aparezca en el archivo FLA.

En esta sección se describe un archivo SWC y se explica cómo importar y exportar archivos SWC en Flash.

Aspectos básicos de los archivos SWC

Un archivo SWC es un archivo similar a zip (empaquetado y ampliado mediante el formato de archivos PKZIP) que genera la herramienta de edición de Flash.

En la tabla siguiente se describe el contenido de un archivo SWC:

Archivo	Descripción
catalog.xml	(Obligatorio) Muestra una lista con el contenido del paquete de componentes y cada uno de sus componentes y funciona como directorio para los otros archivos del archivo SWC.
Archivos de ActionScript (AS)	Si el componente se crea mediante Flash Professional 8, el código fuente es uno o varios archivos de ActionScript que contienen una declaración de clase para el componente. El compilador utiliza el código fuente para la verificación de tipos cuando se amplía un componente. El archivo AS no se compila con la herramienta de edición porque el código de bytes compilado ya se encuentra en el archivo SWF que se implementa. El código fuente puede contener definiciones de clases intrínsecas que no contienen cuerpos de función y sólo se proporcionan para la verificación de tipos.
Archivos SWF	(Obligatorio) Archivos SWF que implementan los componentes. En un solo archivo SWF pueden definirse uno o varios componentes. Si el componente se crea con Flash 8, sólo se exporta un componente por cada archivo SWF.
Archivos SWF de previsualización dinámica	(Opcional) Si se especifican, estos archivos SWF se utilizan para obtener una previsualización dinámica en la herramienta de edición. Si se omiten, los archivos SWF que implementan el componente se utilizan para la previsualización dinámica. El archivo SWF de previsualización dinámica puede omitirse en casi todas las clases; debe incluirse sólo si la apariencia del componente depende de datos dinámicos (por ejemplo, un campo de texto que muestra el resultado de una llamada de servicio Web).

Archivo	Descripción
Archivo SWD	(Opcional) Archivo SWD correspondiente al archivo SWF que se implementa y que permite depurar el archivo SWF. El nombre de archivo siempre es el mismo que el del archivo SWF, pero con la extensión .swd.
Archivo PNG	(Opcional) Archivo PNG que contiene el icono de 18 x 18 de 8 bits por píxel que se utiliza para mostrar un icono de componente en las interfaces de usuario de la herramienta de edición. Si no se proporciona ningún icono, se muestra un icono predeterminado. (Véase “Adición de un icono” en la página 199.)
Archivo SWF del inspector de propiedades	(Opcional) Archivo SWF que se utiliza como el inspector de propiedades personalizado en la herramienta de edición. Si se omite, se muestra al usuario el inspector de propiedades predeterminado.

De forma opcional, puede incluir otros archivos en el archivo SWC una vez lo haya generado en el entorno de Flash. Por ejemplo, es posible que desee incluir un archivo Readme (Léame) o el archivo FLA si desea que los usuarios puedan acceder al código fuente del componente. Para añadir archivos, utilice Macromedia Extension Manager (véase www.macromedia.com/es/exchange/em_download/).

Los archivos SWC se expanden en un único directorio, por lo que cada componente debe tener un nombre de archivo exclusivo para evitar conflictos.

Exportación de archivos SWC

Flash permite exportar archivos SWC mediante la exportación de un clip de película como archivo SWC. Cuando se exporta un archivo SWC, Flash informa de los errores en tiempo de compilación como si estuviera probando una aplicación Flash.

Hay dos razones para exportar un archivo SWC:

- Para distribuir un componente finalizado
- Para realizar pruebas durante el desarrollo.

Exportación de un archivo SWC para un componente finalizado

Puede exportar componentes como archivos SWC que contengan todo el código ActionScript, archivos SWF y otros archivos opcionales necesarios para utilizar el componente.

Para exportar un archivo SWC para un componente finalizado:

1. Seleccione el clip de película del componente en la biblioteca de Flash.
2. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Mac) para abrir el menú contextual Biblioteca.
3. Seleccione Exportar archivo SWC en el menú contextual Biblioteca.
4. Guarde el archivo SWC.

Prueba de un archivo SWC durante el desarrollo

En distintas fases del desarrollo, es aconsejable exportar el componente como un archivo SWC y probarlo en una aplicación. Si exporta el archivo SWC a la carpeta Components de la carpeta Configuration del usuario, puede volver a cargar el panel Componentes sin que sea necesario salir de Flash y reiniciarlo.

Para probar un archivo SWC durante el desarrollo:

1. Seleccione el clip de película del componente en la biblioteca de Flash.
2. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Mac) para abrir el menú contextual Biblioteca.
3. Seleccione Exportar archivo SWC en el menú contextual Biblioteca.
4. Navegue hasta la carpeta Components de la carpeta de configuración del usuario.
Configuration/Components

NOTA

Para más información sobre la ubicación de la carpeta, consulte “Carpetas de configuración instaladas con Flash” en *Primeros pasos con Flash*.

5. Guarde el archivo SWC.
6. Seleccione Volver a cargar en el menú de opciones del panel Componentes.
El componente aparece en el panel Componentes.
7. Arrastre el componente desde el panel Componentes hasta un documento.

Importación de archivos SWC de componente en Flash

Cuando distribuya sus componentes a otros desarrolladores, puede incluir las siguientes instrucciones para que puedan instalarlos y utilizarlos inmediatamente.

Para importar un archivo SWC:

1. Copie el archivo SWC en el directorio Configuration/Components.
2. Reinicie Flash.

El icono del componente debe aparecer en el panel Componentes.

Últimos pasos del desarrollo de componentes

Después de crear el componente y prepararlo para empaquetarlo, puede añadir un icono y una sugerencia. Para asegurarse de que ha completado todos los pasos necesarios, puede consultar también [“Lista de comprobación de desarrollo de componentes” en la página 200.](#)

Adición de un icono

Puede añadir un icono que represente el componente en el panel Componentes del entorno de edición de Flash.

Para añadir un icono del componente:

1. Cree una imagen nueva.
La imagen debe medir 18 píxeles cuadrados y guardarse en formato PNG. Debe ser de 8 bits y con transparencia alfa, y el píxel superior izquierdo debe ser transparente para admitir máscaras.
2. Añada la definición siguiente al archivo de clase de ActionScript del componente antes de la definición de la clase:

```
[IconFile("component_name.png")]
```
3. Añada la imagen al mismo directorio en el que se encuentra el archivo FLA. Cuando exporte el archivo SWC, Flash incluirá la imagen en la raíz del archivo.

Adición de una sugerencia

Las sugerencias aparecen cuando un usuario pasa el ratón por encima del nombre o del icono del componente en el panel Componentes del entorno de edición de Flash.

Puede definir una sugerencia en el cuadro de diálogo Definición de componente. Puede acceder a este cuadro de diálogo desde el menú de opciones Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada) del archivo FLA del componente.

Para añadir una sugerencia al cuadro de diálogo Definición de componente:

1. Con el archivo FLA del componente abierto en Flash, asegúrese de que la Biblioteca esté visible (menú Ventana > Biblioteca).
2. Haga clic en el menú de opciones Biblioteca (Windows: haga clic con el botón derecho del ratón; Mac: haga clic con la tecla Control presionada).
El menú de opciones de Biblioteca se encuentra a la derecha de la barra de título Biblioteca y aparece como un icono de tres líneas con un triángulo hacia abajo.
3. Seleccione la opción Definición de componente.
4. En el cuadro de diálogo Definición de componente, en Opciones, active la casilla de verificación Mostrar en el panel de componentes.
Podrá editar el campo Texto de información sobre herramientas.
5. Introduzca el texto correspondiente a su componente en el campo Texto de información sobre herramientas.
6. Haga clic en Aceptar para guardar los cambios.

Lista de comprobación de desarrollo de componentes

Cuando diseñe un componente, siga las siguientes directrices:

- Intente que el tamaño del archivo sea lo más reducido posible.
- Intente que su componente sea lo más reutilizable posible mediante la generalización de sus funciones.
- Utilice la clase `RectBorder` (`mx.skins.halo.RectBorder`), en lugar de los elementos gráficos, cuando dibuje los bordes de los objetos. (Consulte “Clase `RectBorder`” en *Referencia del lenguaje de componentes*.)
- Utilice la aplicación de aspectos basada en etiquetas.
- Defina las variables `symbolName`, `symbolOwner` y `className`.

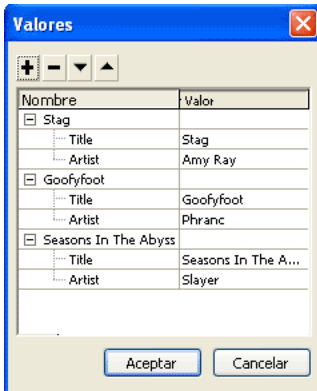
- Asuma un estado inicial. Puesto que las propiedades de estilo se encuentran ahora en el objeto, puede establecer la configuración inicial para los estilos y las propiedades para que su código de inicialización no tenga que establecerlos cuando se cree el objeto, a no ser que el usuario sustituya el estado predeterminado.
- Cuando defina el símbolo no seleccione la opción Exportar en primer fotograma, a no ser que sea absolutamente necesario. Flash carga el componente justo antes de que se utilice en la aplicación Flash, por lo que si selecciona esta opción Flash precargará el componente en el primer fotograma de su elemento principal. La razón por la que habitualmente no se precarga el componente en el primer fotograma tiene que ver con la Web: el componente se carga antes de que se inicie el precargador, lo que inutiliza el objetivo de este último.
- Evite los clips de película de varios fotogramas (excepto en la capa Activos de dos fotogramas).
- Implemente siempre métodos `init()` y `size()` y llame a `Super.init()` y `Super.size()` respectivamente o manténgalos siempre ligeros.
- Evite las referencias absolutas como `_root.myVariable`.
- Utilice `createClassObject()` en lugar de `attachMovie()`.
- Utilice `invalidate()` e `invalidateStyle()` para invocar el método `draw()` en lugar de llamar a `draw()` de forma explícita.
- Cuando incorpore componentes de Flash en su componente, utilice sus símbolos de película sin compilar, que se encuentran en la biblioteca del archivo `StandardComponents.fla` en la carpeta `Configuration/ComponentFLA`.

Cuando cree un nuevo componente personalizado en Macromedia Flash, puede mostrar los valores de propiedades al usuario para que pueda editarlos. Estas propiedades se denominan *propiedades de colección*. El usuario puede editar los valores de propiedades en el cuadro de diálogo Valores que se abre desde un cuadro de texto de la ficha Parámetros del componente.

Los componentes suelen incluir funciones para una tarea específica, al tiempo que proporcionan la flexibilidad necesaria para satisfacer los requisitos de los usuarios de los componentes. Para que un componente sea flexible, a menudo es necesario que sus propiedades también lo sean, lo que significa que las propiedades y los valores de propiedades de algunos componentes pueden ser modificadas por los usuarios de los componentes.

Las propiedades de colección permiten crear un número indeterminado de propiedades editables en un modelo de objetos. Flash proporciona una clase `Collection` para ayudar a administrar tales propiedades mediante el inspector de componentes.

Concretamente, la clase Collection es una clase auxiliar que se utiliza para administrar un grupo de objetos relacionados, cada uno de los cuales se denomina *elemento de la colección*. Si se define una propiedad del componente como un elemento de la colección y se pone a disposición de los usuarios a través del inspector de componentes, éstos podrán añadir, eliminar y modificar los elementos de la colección en el cuadro de diálogo Valores durante la edición.



Las colecciones y elementos de la colección se definen del siguiente modo:

- Defina una propiedad de colección mediante la etiqueta de metadatos Collection en un archivo de ActionScript de un componente. Para más información, consulte [“Etiqueta Collection” en la página 169](#).
- Defina un elemento de la colección como una clase en un archivo de ActionScript independiente que contenga sus propias propiedades Inspectable.

En Flash, las colecciones facilitan la administración de grupos de elementos relacionados mediante programación. (En anteriores versiones de Flash, los creadores de componentes administraban los grupos de elementos relacionados a través de múltiples matrices sincronizadas mediante programación).

Además del cuadro de diálogo Valores, Flash proporciona las interfaces Collection e Iterator para administrar instancias de Collection y valores mediante programación. Consulte [“Interfaz Collection \(sólo en Flash Professional\)”](#) y [“Interfaz Iterator \(sólo en Flash Professional\)”](#) en *Referencia del lenguaje de componentes*.

Este capítulo contiene las siguientes secciones:

Definición de una propiedad de colección	205
Ejemplo sencillo de colección	206
Definición de la clase de un elemento de la colección.....	208
Acceso a información de colección mediante programación.....	209
Exportación de componentes con colecciones a archivos SWC	211
Utilización de un componente que incluye una propiedad de colección	212

Definición de una propiedad de colección

Defina una propiedad de colección mediante la etiqueta `Collection` en un archivo de ActionScript de un componente. Para más información, consulte [“Etiqueta `Collection`” en la página 169](#).

NOTA

En esta sección se supone que sabe cómo crear componentes y propiedades de componente Inspectable.

Para definir una propiedad de colección:

1. Cree un archivo FLA para el componente. Véase [“Creación de un clip de película del componente” en la página 149](#).
2. Cree una clase de ActionScript. Véase [“Creación del archivo de clase de ActionScript” en la página 154](#).
3. En la clase de ActionScript, inserte la etiqueta de metadatos `Collection`. Para más información, consulte [“Etiqueta `Collection`” en la página 169](#).
4. Defina métodos `get` y `set` para la colección en el archivo de ActionScript del componente.
5. Para añadir al archivo FLA las clases de utilidades, seleccione `Ventana > Bibliotecas comunes > Clases` y arrastre `UtilsClasses` a la biblioteca del componente.
`UtilsClasses` contiene el paquete `mx.utils.*` de la interfaz `Collection`.

NOTA

Como `UtilsClasses` se asocia con el archivo FLA y no con la clase de ActionScript, Flash emite errores de compilador cuando se comprueba la sintaxis mientras se visualiza la clase de ActionScript del componente.

6. Programe una clase que contenga las propiedades de elementos de la colección.
Véase [“Definición de la clase de un elemento de la colección” en la página 208](#).

Ejemplo sencillo de colección

A continuación se muestra un ejemplo sencillo de un archivo de clase de componente denominado `MyShelf.as`. Este ejemplo contiene una propiedad de colección junto a un conjunto mínimo de importaciones, métodos y declaraciones para un componente que hereda de la clase `UIObject`.

Si se importa `mx.utils.*` en este ejemplo, no es necesario utilizar los nombres de clase completos de `mx.utils`. Por ejemplo, `mx.utils.Collection` puede escribirse como `Collection`.

```
import mx.utils.*;
// declaración de clase estándar
class MyShelf extends mx.core.UIObject
{
// variables necesarias para todas las clases
    static var symbolName:String = "MyShelf";
    static var symbolOwner:Object = Object(MyShelf);
    var className:String = "MyShelf";

// etiqueta de metadatos Collection y atributos
    [Collection(variable="myCompactDiscs",name="My Compact
    Discs",collectionClass="mx.utils.CollectionImpl",
    collectionItem="CompactDisc", identifier="Title")]

// métodos get y set para la colección
    public function get MyCompactDiscs():mx.utils.Collection
    {
        return myCompactDiscs;
    }
    public function set MyCompactDiscs(myCDs:mx.utils.Collection):Void
    {
        myCompactDiscs = myCDs;
    }

// miembro de clase privado
    private var myCompactDiscs:mx.utils.Collection;

// Es necesario programar una referencia a la clase de elemento de la
// colección
// para obligar al compilador a que la incluya como una dependencia
// en el archivo SWC
    private var collItem:CompactDisc;

// Es necesario programar una referencia a la clase mx.utils.CollectionImpl
// para obligar al compilador a que la incluya como una dependencia
// en el archivo SWC
    private var coll:mx.utils.CollectionImpl;

// métodos necesarios para todas las clases
```

```

function init(Void):Void {
    super.init();
}
function size(Void):Void {
    super.size();
}
}

```

Para crear un archivo FLA para acompañar esta clase con el fin de realizar pruebas:

1. En Flash, seleccione Archivo > Nuevo y cree un documento de Flash.
2. Seleccione Insertar > Nuevo símbolo. Asígnele **MyShelf** como nombre, identificador de vinculación y nombre de clase de AS 2.0.
3. Anule la selección de Exportar en primer fotograma y haga clic en Aceptar.
4. Seleccione el símbolo de MyShelf en la biblioteca y elija Definición de componente en el menú Opciones de Biblioteca. Introduzca el nombre de clase de ActionScript 2.0 **MyShelf**.
5. Seleccione Ventana > Bibliotecas comunes > Clases y arrastre UtilClasses a la biblioteca de MyShelf fla.
6. En la línea de tiempo del símbolo de MyShelf, denomine a una de las capas **Activos**. Cree otra capa y denomínela **Acciones**.
7. Coloque una función `stop()` en el fotograma 1 de la capa Acciones.
8. Seleccione el fotograma 2 de la capa Activos y, a continuación, Insertar > Línea de tiempo > Fotograma clave.
9. Abra el archivo StandardComponents.fla de la carpeta Configuration/ComponentFLA y arrastre una instancia de UIObject al escenario de MyShelf en el fotograma 2 de la capa Activos.

Debe incluir UIObject en el archivo FLA del componente porque, tal y como se aprecia en el archivo de clase anterior, MyShelf amplía UIObject.

10. En el fotograma 1 de la capa Activos, dibuje un estante.
Puede ser simplemente un rectángulo; se trata únicamente de una representación visual del componente MyShelf que se utilizará con fines de aprendizaje.

11. Seleccione el clip de película MyShelf de la biblioteca y, a continuación, seleccione Convertir en clip compilado.

De esta forma podrá arrastrar el archivo SWF MyShelf (el clip compilado que se añade a la biblioteca) al archivo MyShelf.fla para probar el componente. Cuando vuelva a compilar el componente, aparecerá un cuadro de diálogo Solucionar conflicto de biblioteca porque ya existe una versión anterior del componente en la biblioteca. Elija la opción para reemplazar elementos existentes.

NOTA

Debería haber creado la clase CompactDisc; de lo contrario, recibirá errores de compilador al convertir en clip compilado.

Definición de la clase de un elemento de la colección

Las propiedades de un elemento de la colección se programan en una clase de ActionScript independiente, que se define del siguiente modo:

- Defina la clase de forma que no amplíe UIObject o UIComponent.
- Defina todas las propiedades mediante la etiqueta Inspectable.
- Defina todas las propiedades como variables. No utilice métodos `get` ni `set` (captador/definidor).

A continuación se muestra un ejemplo sencillo de un archivo de clase de elemento de la colección denominado CompactDisc.as.

```
class CompactDisc{
    [Inspectable(type="String", defaultValue="Title")]
    var title:String;
    [Inspectable(type="String", defaultValue="Artist")]
    var artist:String;
}
```

Para ver el archivo de clase CompactDisc.as, consulte [“Ejemplo sencillo de colección” en la página 206](#).

Acceso a información de colección mediante programación

Flash proporciona acceso programático a los datos de colección a través de las interfaces `Collection` e `Iterator`. La interfaz `Collection` permite añadir, modificar y eliminar elementos de una colección. La interfaz `Iterator` permite recorrer los elementos de una colección.

Las interfaces `Collection` e `Iterator` se pueden utilizar de dos formas posibles:

- “Acceso a información de colección en un archivo de clase de componente (AS)” en la página 209
- “Acceso a elementos de la colección de una aplicación Flash en tiempo de ejecución” en la página 210

Los desarrolladores avanzados pueden crear, rellenar, eliminar y acceder a colecciones mediante programación; para más información, consulte “Interfaz `Collection` (sólo en Flash Professional)” en *Referencia del lenguaje de componentes*.

Acceso a información de colección en un archivo de clase de componente (AS)

En un archivo de clase de componente se puede escribir código que interactúe con elementos de la colección definidos durante la edición o en tiempo de ejecución.

Se puede acceder a la información de elementos de la colección de un archivo de clase de componente de cualquiera de las siguientes maneras.

- La etiqueta `Collection` incluye un atributo `variable`, para el cual se especifica una variable del tipo `mx.utils.Collection`. Utilice esta variable para acceder a la colección, como se muestra en este ejemplo:

```
[Collection(name="LinkButtons", variable="__linkButtons",
  collectionClass="mx.utils.CollectionImpl", collectionItem="ButtonC",
  identifier="ButtonLabel")]
public var __linkButtons:mx.utils.Collection;
```

- Acceda a la interfaz `Iterator` de elementos de la colección llamando al método `Collection.getIterator()`, como se muestra en este ejemplo:
`var itr:mx.utils.Iterator = __linkButtons.getIterator();`

- Utilice la interfaz Iterator para pasar por los elementos de la colección. El método `Iterator.next()` devuelve un objeto, por lo que es necesario definir el tipo del elemento de la colección, como se muestra en este ejemplo:

```
while (itr.hasNext()) {  
    var button:ButtonC = ButtonC(itr.next());  
    ...  
}
```

- Acceda a las propiedades de elementos de la colección apropiadas para la aplicación, como se muestra en este ejemplo:

```
item.label = button.ButtonLabel;  
  
if (button.ButtonLink != undefined) {  
    item.data = button.ButtonLink;  
}  
else {  
    item.enabled = false;  
}
```

Acceso a elementos de la colección de una aplicación Flash en tiempo de ejecución

Si una aplicación Flash utiliza un componente que tiene una propiedad de colección, puede acceder a ésta en tiempo de ejecución. En este ejemplo se añaden varios elementos a una propiedad de colección mediante el cuadro de diálogo Valores y se muestran en tiempo de ejecución mediante las API Collection e Iterator.

Para acceder a elementos de la colección en tiempo de ejecución:

1. Abra el archivo `MyShelf.fla` que creó anteriormente.
Véase [“Ejemplo sencillo de colección” en la página 206](#).
Este ejemplo se basa en el componente `MyShelf` y la colección `CompactDisc`.
2. Abra el panel Biblioteca, arrastre el componente al escenario y asígnele un nombre de instancia.
En este ejemplo se utiliza el nombre de instancia `myShelf`.
3. Seleccione el componente, abra el inspector de componentes y abra la ficha Parámetros. Haga clic en la línea que contiene la propiedad de colección y, a continuación, en el icono de lupa situado a la derecha de la línea. Flash muestra el cuadro de diálogo Valores.
4. Utilice el cuadro de diálogo Valores para introducir valores en la propiedad de colección.

5. Con el componente seleccionado en el escenario, abra el panel Acciones e introduzca el siguiente código (que debe asociarse con el componente):

```
onClipEvent (mouseDown) {
    import mx.utils.Collection;
    import mx.utils.Iterator;
    var myColl:mx.utils.Collection;
    myColl = _parent.myShelf.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;
        trace("Title: " + title + " Artist: " + artist);
    }
}
```

Para acceder a una colección, utilice la sintaxis *componentName.collectionVariable*; para acceder a un repetidor y pasar por los elementos de la colección, utilice *componentName.collectionVariable.getIterator()*.

6. Seleccione Control > Probar película y haga clic en la representación gráfica del estante para ver los datos de la colección en el panel Salida.

Exportación de componentes con colecciones a archivos SWC

Cuando se distribuye un componente que tiene una colección, el archivo SWC debe contener los siguientes archivos dependientes:

- Interfaz Collection
- Clase de implementación de la colección
- Clase de elemento de la colección
- Interfaz Iterator

De estos archivos, suelen utilizarse en el código las interfaces Collection e Iterator, que se marcan como clases dependientes. Flash incluye de forma automática archivos dependientes en el archivo SWC y el archivo SWF de salida.

Sin embargo, la clase de implementación de la colección (*mx.utils.CollectionImpl*) y la clase de elemento de la colección específica del componente no se incluyen automáticamente en el archivo SWC.

Para incluir la clase de implementación de la colección y la clase de elemento de la colección en el archivo SWC, es necesario definir variables privadas en el archivo de ActionScript del componente, como se muestra en el siguiente ejemplo:

```
// clase de elemento de la colección
private var collItem:CompactDisc;
// clase de implementación de la colección
private var coll:mx.utils.CollectionImpl;
```

Para más información sobre archivos SWC, consulte [“Aspectos básicos de los archivos SWC” en la página 196.](#)

Utilización de un componente que incluye una propiedad de colección

Cuando se utiliza un componente que incluye una propiedad de colección, normalmente se utiliza el cuadro de diálogo Valores para establecer los elementos de la colección.

Para utilizar un componente que incluye una propiedad de colección:

1. Añada el componente al escenario.
2. Utilice el inspector de propiedades para asignar un nombre a la instancia del componente.
3. Abra el inspector de componentes y seleccione la pestaña Parámetros.
4. Haga clic en la línea que contiene la propiedad de colección y, a continuación, en el icono de lupa situado a la derecha de la línea.
Flash muestra el cuadro de diálogo Valores.
5. Haga clic en el botón Añadir (+) y defina un elemento de la colección.
6. Haga clic en el botón Añadir (+), Eliminar (-) y los botones de flecha para añadir, mover y eliminar elementos de la colección.
7. Haga clic en Aceptar.

Para más información sobre cómo acceder a la colección mediante programación, consulte [“Acceso a elementos de la colección de una aplicación Flash en tiempo de ejecución” en la página 210.](#)

Índice alfabético

A

- accesibilidad 22
- ActionScript, archivos de clase 154
- actualizar componentes de la versión 1 68
- ampliar clases 148
- aplicar aspectos a componentes 102
- archivo de clase
 - ejemplo 154, 206
 - información 156
- archivo de clase del componente. *Véase* archivo de clase aspectos
 - aplicar a componentes 108
 - aplicar a subcomponentes 109
 - crear variables para 181
 - editar 105
 - identificadores de vinculación
 - Véase también nombres de componentes individuales*

B

- biblioteca
 - Biblioteca, panel 58
 - clips compilados de la 58
 - StandardComponents 150

C

- caché de mapa de bits no compatible 20
- captador/definidor, métodos 160
- cargar, componentes 67
- clase principal, seleccionar 146

clases

- ampliar 148
- crear referencias a (tutorial) 27
- crear. *Véase* crear componentes
- definir 158
- importar 157
- seleccionar una clase principal 146
- UIComponent 147
- UIObject 147
- y herencia de componente 19

- class, palabra clave 158
- className, variable 158

- clips compilados
 - del panel Biblioteca 58
 - información 21
- clips de película
 - crear 149
 - definir como componente 152
- colección, elemento 208
- colección, propiedades
 - acceder mediante programación 209
 - definir 205
 - definir clases 208
 - ejemplo 206
 - exportar componentes 211
 - utilizar 212
- Collection, etiqueta 169
- colores
 - establecer propiedades de estilo 97
 - personalizar 86
- columnas
 - añadir (tutorial) 34

- componentes 154
 - ampliar clases 148
 - añadir a documentos de Flash 54
 - añadir en tiempo de ejecución 56
 - añadir sugerencias 200
 - añadir un icono 199
 - archivos de origen 133
 - arquitectura 18
 - asignar aspectos 181
 - captador/definidor, métodos 160
 - cargar 67
 - categorías, descripción 17
 - clase de ActionScript 154
 - className, variable 158
 - compatibilidad con Flash Player 18
 - crear clips de película 149
 - definir el método draw() 177
 - definir el método init() 174
 - definir el método size() 177
 - definir parámetros 171
 - definir variables 159
 - disponibles en las ediciones de Flash MX 12
 - distribuir eventos 179
 - editar clips de película 150
 - ejemplo de archivo de clase 154
 - ejemplo de archivo de clase con colección 206
 - ejemplo de creación de un componente 136, 185
 - eliminar 62
 - estilos 182
 - estructura de 134
 - etiquetas de metadatos
 - eventos 69
 - eventos comunes 180
 - exportar archivos SWC 197
 - exportar componente como archivo SWC 198
 - exportar y distribuir 195
 - herencia 19
 - importar archivos SWC 199
 - información general sobre la clase 156
 - instalar 12
 - invalidación, información 178
 - lista de comprobación de desarrollo 200
 - metadatos, etiqueta ComponentTask 170
 - precargar 65
 - previsualizar 65
 - probar archivos SWC 198
 - registrar aspectos en estilos 182
 - seleccionar nombres de símbolo 158
 - seleccionar una clase principal 146

- symbolOwner, variable 158
 - utilizar en una aplicación (tutorial) 23
 - Véase también nombres de componentes individuales*
- componentes de la versión 1, actualizar 68
- Componentes, panel 54
- components
 - creating subobjects 174
- ComponentTask, etiqueta
 - JavaScript (JSFL) 170
- convenciones tipográficas 9
- createClassObject() method 174
- CSSStyleDeclaration 91, 92
- cuadrículas de datos. *Véase* DataGrid, componente
- cuadrículas. *Véase* DataGrid, componente

D

- DataGrid, componente
 - añadir columnas (tutorial) 34
 - vinculación a DataSet (tutorial) 31
- DataSet, componente, vinculación a XMLConnector y DataGrid (tutorial) 31
- declaraciones de estilo
 - clase predeterminada 94
 - definir clase 94
 - global 91
 - personalizado 92
- declaraciones de estilo global 91
- defaultPushButton, propiedad 63
- Delegate, clase (tutorial) 78
- DepthManager, clase, información general 64
- destinatarios de este documento 8
- detectores
 - ámbito 77
 - funciones 75
 - información 70
 - objetos 71
 - utilizar con componentes (tutorial) 35
 - utilizar con instancias de componentes (tutorial)
- detectores de eventos. *Véase* detectores
- Dial, componente 136, 185
- difusor 70
- distribuidor (difusor de eventos) 70
- distribuir eventos 179
- documentación
 - guía de términos 9
 - información general 8
- draw(), método, definir 177

E

- ediciones de Flash MX y componentes disponibles 12
- eliminar componentes 62
- escala de 9 divisiones no compatible 20
- estilos
 - crear componentes 182
 - definición global 91
 - definir en una instancia 89
 - definir personalizados 92
 - determinar la prioridad 97
 - establecer 86
 - información 86
 - utilizar (tutorial) 30
 - Véase también nombres de componentes individuales*
- etiquetas. *Véase* metadatos
- Event, etiqueta de metadatos 165
- eventos
 - comunes 180
 - delegar ámbito 78
 - distribuir 179
 - funciones de controlador 69
 - información 69
 - metadatos 165
 - objeto de evento 82
 - Véase también nombres de componentes individuales*
- exportar componentes 195

F

- Flash Player
 - compatibilidad 68
 - y componentes 18
- FlashType no compatible 20
- FocusManager, clase, crear desplazamiento de la selección 63
- funciones de controlador 69

H

- Halo, tema 115
- handleEvent, función callback 74
- herencia, en componentes de la versión 2 19
- hojas de estilos
 - clase 86
 - personalizadas 86
- hojas de estilos de clase 86

I

- icono, de un componente 199
- identificadores de vinculación para aspectos 102
- import, sentencia 157
- init(), método, definir 174
- Inspectable, parámetros 163
- inspector de componentes
 - configurar parámetros 59
 - Vinculaciones, ficha 32
- inspector de propiedades 59
- instalar componentes 12
- instancias
 - declaraciones de estilo 86
 - definir estilos 89
- invalidate(), método 178

J

- JavaScript de Flash (JSFL), etiqueta ComponentTask 170

L

- Label, componente tutorial 43
- lectores de pantalla, accesibilidad 22

M

- metadatos
 - Collection, etiqueta 169
 - ComponentTask, etiqueta 170
 - etiquetas, lista de 162
 - Event, etiqueta 165
 - información
 - Inspectable, etiqueta 163
- MovieClip, clase, ampliar 148

O

- on(), controlador de eventos 83

P

- paquetes 18
- parámetros de componentes
 - definir 171
 - establecer 59
 - información 59
 - Inspectable 163
 - visualizar 59
 - Véase también nombres de componentes individuales*
- parámetros. *Véase* parámetros de componentes
- personalizar
 - texto 86
- personalizar color y texto, usar hojas de estilos 86
- prácticas recomendadas para el desarrollo de componentes 200
- precargar componentes 65
- Previsualización dinámica, característica 65
- previsualizar componentes 65
- probar archivos SWC 198
- propiedades de aspecto
 - cambiar en el prototipo 113
 - establecer 102
- propiedades de estilo, color 97
- prototipo 113

R

- recursos, adicionales de Macromedia 9
- requisitos del sistema para componentes 8

S

- Sample, tema 115
- ScrollPane, componente
 - tutorial 43
- servicio Web, conectar con (tutorial) 30
- size(), método, definir 177
- StandardComponents, biblioteca 150
- subclases, utilizar para sustituir aspectos 113
- subcomponentes, aplicar aspectos 109
- subjects, creating 174
- sugerencias para el código, activar 62
- superclass, palabra clave 158
- SWC, archivos
 - exportar 197
 - exportar propiedades de colección 211
 - formato de archivo 196
 - importar 199
 - información 21

- probar 198
 - y clips compilados 21
- symbolName, variable 158
- symbolOwner, variable 158

T

- tabulación 63
- temas
 - aplicar 121
 - crear 119
 - información 115
- términos utilizados en la documentación 9
- TextInput, componente (tutorial) 43
- texto de información, añadir 200
- texto, personalizar 86
- tipos de datos, asignar a instancias (tutorial) 29

U

- UIComponent, clase
 - información general 147
 - y herencia de componente 19
- UIObject, clase
 - información 147

V

- Valores, cuadro de diálogo 204
- variables, definir 159
- versión 2, componentes
 - herencia 19
 - ventajas 16
 - y Flash Player 18
- vinculación de datos, con archivo XML (tutorial) 31
- Vinculaciones, ficha, en la aplicación de ejemplo (tutorial) 32

W

- WebService, clase (tutorial) 30

X

- XMLConnector, componente
 - cargar un archivo XML externo (tutorial) 34
 - especificar esquema (tutorial) 32
 - vinculación al componente DataSet (tutorial) 31