



8

Marchi

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 e WebHelp sono marchi registrati o marchi di Macromedia, Inc. e possono essere registrati negli Stati Uniti o presso altre giurisdizioni, anche a livello internazionale. Altri nomi di prodotti, logo, disegni, titoli, parole o frasi citati in questa pubblicazione possono essere marchi registrati, marchi di servizio o nomi commerciali di Macromedia, Inc. o di altre società e possono essere registrati in alcune giurisdizioni, anche a livello internazionale.

Informazioni su terze parti

Questo manuale contiene collegamenti a siti Web di terze parti che non sono sotto il controllo di Macromedia. Macromedia non potrà quindi essere ritenuta responsabile per il contenuto di qualsiasi sito collegato. Qualora si decida di accedere a un sito Web di terze parti menzionato nel presente documento, lo si farà sotto la propria completa responsabilità e a proprio rischio. Macromedia fornisce questi collegamenti solo per comodità dell'utente e l'inclusione del collegamento non implica che Macromedia sottoscriva o accetti qualsiasi responsabilità per il contenuto di tali siti di terze parti.

Tecnologia per la compressione e la decompressione vocale concessa in licenza da Nellymoser, Inc. (www.nellymoser.com).



Tecnologia per la compressione e la decompressione video Sorenson™ Spark™, concessa in licenza da Sorenson Media, Inc.

Browser Opera® Copyright © 1995-2002 di Opera Software ASA e dei suoi fornitori. Tutti i diritti riservati.

Il video Macromedia Flash 8 è basato sulla tecnologia video On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Tutti i diritti riservati. <http://www.on2.com>.

Visual SourceSafe è un marchio registrato o un marchio di Microsoft Corporation negli Stati Uniti e/o in altri PAESI.

Copyright © 2005 Macromedia, Inc. Tutti i diritti riservati. Nessuna parte del presente manuale può essere copiata, fotocopiata, riprodotta, tradotta o convertita in qualsiasi formato elettronico o meccanico senza la previa autorizzazione scritta di Macromedia, Inc. Nonostante quanto sopra specificato, il proprietario o l'utente autorizzato del software con cui questo manuale è stato fornito è autorizzato a stampare una copia di questo manuale da una versione elettronica dello stesso con il solo scopo di apprendere l'utilizzo del suddetto software, a condizione che nessuna porzione di questo manuale venga stampata, riprodotta, distribuita, rivenduta o trasmessa per qualsiasi altro scopo, compresi a titolo esemplificativo i fini commerciali quali la vendita di copie di questa documentazione o la fornitura di servizi di assistenza a pagamento.

Riconoscimenti

Responsabili progetto: Sheila McGinn

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

Caporedattore: Rosana Francescato

Lead Editor: Lisa Stanziano

Redazione: Mary Ferguson, Mary Kraemer, Lisa Stanziano

Responsabile produzione: Patrice O'Neill, Kristin Conradi, Yuko Yagi

Produzione e progetto multimediale: Adam Barnett, Aaron Begley, Paul Benkman. John Francis, Geeta Karmarkar, Masayo Noda, Paul Rangel, Arena Reed, Mario Reynoso

Uno speciale ringraziamento a Jody Bleye, Mary Burger, Lisa Friendly, Stephanie Gowin, Bonnie Loo, Nivesh Rajbhandari, Mary Ann Walsh, Erick Vera, ai beta tester e a tutti i team engineering e QA di Flash e Flash Player.

Prima edizione: Settembre 2005

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

Indice

Introduzione	7
A chi è destinato questo manuale	8
Requisiti di sistema	8
Informazioni sulla documentazione	8
Convenzioni tipografiche	9
Termini utilizzati in questo manuale	9
Altro materiale di riferimento	9
 Capitolo 1: Informazioni sui componenti.	11
Installazione dei componenti	12
Percorso di memorizzazione dei file dei componenti	14
Modifica dei file dei componenti	14
Vantaggi dell'uso dei componenti	16
Categorie di componenti	16
Informazioni sull'architettura dei componenti della versione 2.	17
Funzioni dei componenti della versione 2.	19
Informazioni su file SWC e clip compilati	20
Accessibilità e componenti	21
 Capitolo 2: Creazione di un'applicazione con i componenti (solo Flash Professional).	23
Informazioni sull'esercitazione Fix Your Mistake (Fatti perdonare) .	23
Creazione della pagina principale	25
Associazione dei componenti dati per visualizzare le idee regalo . . .	32
Visualizzazione dei dettagli dei regali.	37
Creazione della schermata Checkout	42
Prova dell'applicazione	51
Visualizzazione dell'applicazione completa	51

Capitolo 3: Operazioni con i componenti	53
Pannello Componenti	54
Aggiunta di componenti ai documenti Flash	54
Componenti del pannello Libreria	58
Impostazione dei parametri dei componenti	59
Ridimensionamento dei componenti	61
Eliminazione dei componenti dai documenti Flash	62
Uso dei suggerimenti sul codice	62
Navigazione con attivazione personalizzata del componente	62
Gestione della profondità dei componenti in un documento	64
Componenti in Anteprima dal vivo	64
Uso di un precaricatore con i componenti	65
Informazioni sul caricamento dei componenti	66
Aggiornamento dei componenti della versione 1 all'architettura della versione 2	67
 Capitolo 4: Gestione degli eventi dei componenti	 69
Uso dei listener per la gestione degli eventi	70
Delega di eventi	79
Informazioni sull'oggetto evento	82
Uso del gestore di eventi on()	83
 Capitolo 5: Personalizzazione dei componenti	 85
Uso degli stili per personalizzare il testo e il colore dei componenti	86
Informazioni sull'associazione di skin ai componenti	101
Informazioni sui temi	115
Combinazione di skin e di stili per personalizzare un componente	125
 Capitolo 6: Creazione dei componenti	 133
File di origine dei componenti	134
Panoramica sulla struttura dei componenti	134
Creazione del primo componente	135
Selezione di una classe principale	145
Creazione del clip filmato di un componente	149
Creazione del file di classe ActionScript	154
Come incorporare componenti esistenti in un componente	185
Esportazione e distribuzione di un componente	194
Passaggi finali dello sviluppo di un componente	198

Capitolo 7: Proprietà della raccolta	201
Definizione della proprietà di una raccolta	203
Semplice esempio di raccolta	203
Definizione della classe per una voce della raccolta.	206
Accesso alle informazioni della raccolta a livello di codice	206
Esportazione di componenti che dispongono di raccolte in un file SWC	209
Uso di un componente che dispone di una proprietà della raccolta	210
Indice analitico	211

Introduzione

Macromedia Flash Basic 8 e Macromedia Flash Professional 8 sono gli strumenti standard per la creazione di codice in grado di offrire esperienze Web di grande impatto. I componenti sono gli elementi strutturali di base delle applicazioni Internet che assicurano tali esperienze. I *componenti* sono clip filmato dotati di parametri, impostati durante la fase di creazione in Macromedia Flash, e di metodi, proprietà ed eventi ActionScript che ne consentono la personalizzazione in fase di runtime. I componenti sono concepiti per consentire agli sviluppatori di riutilizzare e condividere il codice, nonché di incorporare funzionalità complesse che i designer possono usare e personalizzare senza avvalersi di ActionScript.

I componenti sono basati sulla versione 2 di Macromedia Component Architecture, che consente di creare in modo facile e rapido applicazioni robuste con un aspetto e un comportamento omogenei. In questo manuale viene illustrato come creare applicazioni con componenti della versione 2. La *Guida di riferimento dei componenti* illustra l'interfaccia API (Application Programming Interface) di ogni componente. Vengono proposti scenari di uso ed esempi di procedure da utilizzare con i componenti Flash della versione 2, oltre alle descrizioni delle API riportate in ordine alfabetico.

È possibile usare i componenti creati da Macromedia, scaricare i componenti creati da altri sviluppatori oppure creare autonomamente nuovi componenti.

Questo capitolo contiene le seguenti sezioni:

A chi è destinato questo manuale	8
Requisiti di sistema	8
Informazioni sulla documentazione	8
Convenzioni tipografiche	9
Termini utilizzati in questo manuale	9
Altro materiale di riferimento	9

A chi è destinato questo manuale

La pubblicazione è rivolta agli sviluppatori che creano applicazioni Flash e desiderano utilizzare i componenti per accelerare la fase di sviluppo. È necessario disporre di una buona conoscenza dello sviluppo di applicazioni in Flash e della creazione di script ActionScript.

Se non si è molto esperti nella creazione di script ActionScript, è possibile aggiungere i componenti a un documento, impostare i relativi parametri nella finestra di ispezione Proprietà o nella finestra di ispezione dei componenti e gestirne gli eventi nel pannello Comportamenti. Ad esempio, è possibile associare un comportamento Vai alla pagina Web a un componente Button che apre un URL in un browser Web quando si fa clic sul pulsante senza scrivere alcun codice ActionScript.

Se invece si desidera creare applicazioni più complesse, è possibile creare i componenti in modo dinamico, usare ActionScript per impostare le proprietà e chiamare i metodi in fase di runtime, utilizzando il modello di evento listener per gestire gli eventi.

Per ulteriori informazioni, vedere [Capitolo 3, “Operazioni con i componenti”](#) a pagina 53.

Requisiti di sistema

I componenti Macromedia non presentano ulteriori requisiti di sistema rispetto a Flash.

Eventuali file SWF che utilizzino i componenti della versione 2 devono essere visualizzati con Flash Player 6 (6.0.79.0) o successivo e devono essere pubblicati per ActionScript 2.0 (è possibile eseguire tale impostazione tramite File > Impostazioni di pubblicazione, nella scheda Flash).

Informazioni sulla documentazione

In questa pubblicazione è illustrato in dettaglio l'uso dei componenti per sviluppare le applicazioni Flash. È necessario disporre di un'adeguata conoscenza di Macromedia Flash e ActionScript. La documentazione specifica su Flash e i prodotti correlati è disponibile separatamente.

Questo documento è disponibile sotto forma di file PDF e di Guida in linea. Per visualizzare la Guida in linea, avviare Flash e selezionare ? > Uso dei componenti.

Per informazioni su Macromedia Flash, consultare i seguenti documenti:

- *Uso di Flash*
- *Apprendimento di ActionScript 2.0 in Flash*
- *Guida di riferimento di ActionScript 2.0*
- *Guida di riferimento dei componenti*

Convenzioni tipografiche

In questa pubblicazione vengono usate le seguenti convenzioni tipografiche:

- *Carattere corsivo*: indica un valore che deve essere sostituito dall'utente (ad esempio, il percorso di una cartella).
- *Carattere codice*: indica il codice ActionScript, inclusi nomi dei metodi e delle proprietà.
- *Carattere codice corsivo*: indica un elemento di codice che deve essere sostituito, ad esempio un parametro ActionScript.
- **Carattere grassetto**: indica un valore da immettere.

Termini utilizzati in questo manuale

In questo manuale vengono utilizzati i termini seguenti:

In (fase di) runtime Durante l'esecuzione del codice in Flash Player.

Durante la (fase di) creazione Nel corso delle operazioni eseguite nell'ambiente di creazione Flash.

Altro materiale di riferimento

Per le informazioni più recenti su Flash, consigli di utenti più esperti, argomenti avanzati, esempi, suggerimenti e aggiornamenti, visitare il sito Web Centro per sviluppatori Macromedia www.macromedia.com/it/devnet, che viene aggiornato periodicamente. Per reperire le ultime notizie su Flash e usare al meglio il programma, visitare periodicamente il sito.

Per le note tecniche, gli aggiornamenti della documentazione e i collegamenti ad altro materiale di riferimento nella comunità di utenti Flash, consultare il Centro di assistenza Macromedia Flash all'indirizzo www.macromedia.com/support/flash.

Per informazioni dettagliate sui termini, la sintassi e l'uso di ActionScript, consultare i manuali *Apprendimento di ActionScript 2.0 in Flash* e *Guida di riferimento di ActionScript 2.0*.

Per un'introduzione all'uso dei componenti, vedere il seminario Macromedia On Demand, Using UI Components all'indirizzo www.macromedia.com/macromedia/events/online/ondemand/index.html.

I componenti di Macromedia Flash sono clip filmato dotati di parametri che consentono di modificarne l'aspetto e il comportamento. Un componente può essere un semplice controllo dell'interfaccia utente, ad esempio un pulsante di scelta o una casella di controllo, oppure può includere un contenuto, come nel caso di un riquadro di scorrimento; inoltre, un componente può anche non essere visualizzabile, come FocusManager che consente di controllare gli oggetti che vengono attivati in un'applicazione.

Grazie ai componenti, è possibile creare applicazioni Macromedia Flash complesse, anche in mancanza di una conoscenza approfondita di ActionScript. Anziché creare pulsanti, caselle combinate ed elenchi personalizzati, è infatti possibile trascinare questi componenti dal pannello Componenti per aggiungere funzionalità alle applicazioni. Anche l'aspetto dei componenti può essere facilmente personalizzato per soddisfare qualunque esigenza di progettazione.

I componenti sono basati sulla versione 2 di Macromedia Component Architecture, che consente di creare in modo facile e rapido applicazioni robuste con un aspetto e un comportamento omogenei. L'architettura della versione 2 include classi sulle quali sono basati tutti i componenti, meccanismi di skin e stili che consentono di personalizzare l'aspetto dei componenti, un modello di evento broadcaster/listener, la gestione della profondità e dell'attivazione, l'implementazione dell'accessibilità e altro ancora.

NOTA

Alla pubblicazione dei componenti della versione 2, è necessario impostare le impostazioni di pubblicazione per la pubblicazione per ActionScript 2.0 (File > Impostazioni di pubblicazione, scheda Flash). I componenti della versione 2 non verranno eseguiti correttamente se pubblicati utilizzando ActionScript 1.0.

Per ogni componente esistono parametri predefiniti che è possibile impostare durante la fase di creazione in Flash. Tutti i componenti dispongono anche di una serie univoca di metodi, proprietà ed eventi di ActionScript, detta anche *API* (Application Programming Interface), che consente di impostare i parametri e altre opzioni in fase di runtime.

Per un elenco completo dei componenti inclusi in Flash Basic 8 e Flash Professional 8, vedere “Installazione dei componenti” a pagina 12. È inoltre possibile scaricare componenti creati da membri della comunità Flash da Macromedia Exchange all'indirizzo www.macromedia.com/cfusion/exchange/index.cfm.

Questo capitolo contiene le seguenti sezioni:

Installazione dei componenti	12
Percorso di memorizzazione dei file dei componenti	14
Modifica dei file dei componenti	14
Vantaggi dell'uso dei componenti	16
Categorie di componenti	16
Informazioni sull'architettura dei componenti della versione 2	17
Funzioni dei componenti della versione 2	19
Informazioni su file SWC e clip compilati	20
Accessibilità e componenti	21

Installazione dei componenti

Una serie di componenti Macromedia è già installata quando si avvia Flash per la prima volta. Il pannello Componenti riporta quelli installati.

In Flash Basic 8 sono disponibili i seguenti componenti:

- 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

In Flash Professional 8 sono disponibili, oltre ai componenti Flash Basic 8, le classi e i componenti aggiuntivi seguenti:

- Componente Accordion (solo Flash Professional)
- Componente Alert (solo Flash Professional)
- Classi di associazione dei dati (solo Flash Professional)
- Componente DateField (solo Flash Professional)
- Componente DataGrid (solo Flash Professional)
- Componente DataHolder (solo Flash Professional)
- Componente DataSet (solo Flash Professional)
- Componente DateChooser (solo Flash Professional)
- Componente FLVPlayback (solo Flash Professional)
- Classe Form (solo Flash Professional)
- Componenti Media (solo Flash Professional)
- Componente Menu (solo Flash Professional)
- Componente MenuBar (solo Flash Professional)
- Componente RDBMSResolver (solo Flash Professional)
- Classe Screen (solo Flash Professional)
- Classe Slide (solo Flash Professional)
- Componente Tree (solo Flash Professional)
- Componente WebServiceConnector (solo Flash Professional)
- Componente XMLConnector (solo Flash Professional)
- Componente XUpdateResolver (solo Flash Professional)

Per visualizzare i componenti Flash Basic 8 o Flash Professional 8:

1. Avviare Flash.
2. Scegliere Finestra > Componenti per aprire il pannello Componenti nel caso in cui non sia già visualizzato.
3. Selezionare Interfaccia utente per espandere l'albero e visualizzare i componenti installati.

È anche possibile scaricare i componenti da Macromedia Exchange all'indirizzo www.macromedia.com/it/exchange. Per installare i componenti scaricati da Macromedia Exchange, è necessario scaricare e installare Macromedia Extension Manager dall'indirizzo www.macromedia.com/it/exchange/em_download/.

I componenti vengono visualizzati nel pannello Componenti di Flash. Attenersi alla procedura riportata di seguito per installare i componenti su un computer Windows o Macintosh.

Per installare i componenti in un computer Windows o Macintosh:

1. Uscire da Flash.
2. Inserire il file SWC o FLA che contiene il componente nella cartella seguente sul disco rigido:
 - In Windows: C:\Programmi\Macromedia\Flex 8\lingua\Configuration\Components
 - In Macintosh: Macintosh HD/Applicazioni/Macromedia Flex 8/Configuration/Components (Macintosh)
3. Avviare Flex.
4. Selezionare Finestra > Componenti per visualizzare il componente nel pannello Componenti, nel caso in cui non fosse già visualizzato.

Percorso di memorizzazione dei file dei componenti

I componenti Flex vengono memorizzati nella cartella Configuration dell'applicazione.

NOTA

Per informazioni su queste cartelle, vedere “Cartelle di configurazione installate con Flex” in *Guida introduttiva di Flex*.

I componenti vengono installati nei seguenti percorsi:

- Windows 2000 o Windows XP: C:\Programmi\Macromedia\Flex 8\lingua\Configuration\Components
- Mac OS X: Macintosh HD/Applicazioni/Macromedia Flex 8/Configuration/Components

Modifica dei file dei componenti

I file ActionScript di origine per i componenti sono disponibili in:

- Windows 2000 o Windows XP: C:\Programmi\Macromedia\Flex 8\lingua\First Run\Classes\mx
- Mac OS X: Macintosh HD/Applicazioni/Macromedia Flex 8/First Run/Classes/mx

I file nella directory First Run vengono copiati nel percorso all'interno di Documents and Settings al primo avvio di Flash. I percorsi all'interno di Documents and Settings sono i seguenti:

- Windows 2000 o Windows XP: C:\Documents and Settings*nomeutente*\Impostazioni locali\Dati applicazioni\Macromedia\Flex 8\lingua\Configuration\Classes\mx.
- Mac OS X: *Nomeutente*/Library/Supporto Applicazioni/Macromedia/Flex/8/lingua/Configuration/Classes/mx

All'avvio di Flash, se nel percorso all'interno di Document and Settings non è presente un file, Flash lo copia dalla directory First Run nel percorso all'interno di Documents and Settings.

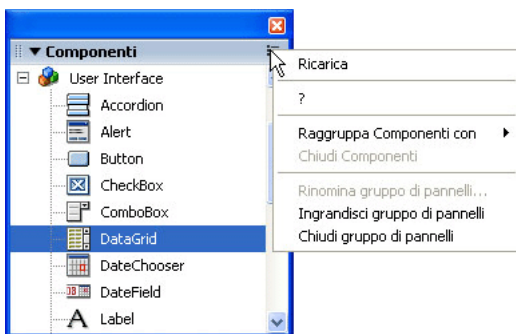
NOTA

Se si desidera modificare i file ActionScript di origine, modificare quelli nel percorso all'interno di Documents and Settings. Se una delle modifiche "interrompe" un componente, Flash ripristinerà la funzionalità originale alla chiusura e successivo riavvio di Flash copiando il file funzionale dalla directory First Run. Tuttavia, se si modificano i file nella directory First Run e questo determina l'interruzione di un componente, potrebbe essere necessario reinstallare Flash per ripristinare i file di origine funzionali.

Se sono stati aggiunti componenti, aggiornare il pannello Componenti.

Per aggiornare il contenuto del pannello Componenti:

- Selezionare Ricarica dal menu del pannello Componenti.



Per rimuovere un componente dal pannello Componenti:

- Rimuovere il file MXP o FLA dalla cartella Configuration.

Vantaggi dell'uso dei componenti

I componenti consentono di separare la procedura di progettazione dell'applicazione dalla procedura di codifica e di riutilizzare parti di codice nei componenti creati autonomamente o in quelli creati da altri sviluppatori che sono stati scaricati e installati.

I componenti consentono ai codificatori di creare funzionalità che i designer possono utilizzare nelle applicazioni. Da una parte, gli sviluppatori possono incorporare nei componenti le funzionalità utilizzate di frequente, dall'altra i designer possono personalizzare l'aspetto e il comportamento dei componenti modificandone i parametri nella finestra di ispezione Proprietà o nella finestra di ispezione dei componenti.

Macromedia Exchange consente agli sviluppatori di Flash di scambiare i componenti nel sito www.macromedia.com/go/exchange. Usando i componenti, non è più necessario creare da zero ogni elemento di un'applicazione Web complessa. È possibile trovare i componenti necessari e riunirli in un documento Flash per creare una nuova applicazione.

I componenti basati sull'architettura della versione 2 condividono funzionalità di base quali stili, gestione degli eventi, meccanismo di skin, gestione dell'attivazione e gestione della profondità. Quando si aggiunge il primo componente della versione 2 a un'applicazione, vengono aggiunti al documento circa 25 KB che forniscono questa funzionalità di base. Quando si aggiungono altri componenti, gli stessi 25 KB vengono riutilizzati anche per i componenti aggiunti. In questo modo le dimensioni del documento aumentano in maniera minore rispetto al previsto. Per informazioni sull'aggiornamento dei componenti, vedere [“Aggiornamento dei componenti della versione 1 all'architettura della versione 2” a pagina 67](#).

Categorie di componenti

I componenti inclusi in Flash sono suddivisi nelle cinque categorie seguenti (tra parentesi vengono indicati i percorsi dei file ActionScript di origine corrispondenti in linea di massima alle categorie):

■ Componenti di dati (mx.data.*)

I componenti dati consentono di caricare e gestire le informazioni dalle varie sorgenti di dati; a questa categoria appartengono i componenti WebServiceConnector e XMLConnector.

NOTA

I file di origine dei componenti di dati non vengono installati con Flash. Vengono invece installati alcuni file ActionScript di supporto.

- Componente FLVPlayback (mx.video.FLVPlayback)
Il componente FLVPlayback consente di includere con facilità un lettore video in un'applicazione Flash per la riproduzione di video in streaming progressivo via HTTP, da un servizio FVSS (Flash Video Streaming Service) o FCS (Flash Communication Server).
- Componenti per contenuti multimediali (mx.controls.*)
I componenti per contenuti multimediali consentono di riprodurre e controllare i media in streaming; MediaController, MediaPlayer e MediaDisplay sono componenti per contenuti multimediali.
- Componenti dell'interfaccia utente (mx.controls.*)
I componenti dell'interfaccia utente, o componenti UI, consentono di interagire con un'applicazione; ad esempio, i componenti RadioButton, CheckBox e TextInput rientrano in questa categoria.
- Gestori (mx.managers.*)
I gestori sono componenti non visualizzabili che consentono di gestire una funzione in un'applicazione, ad esempio l'attivazione o la profondità; FocusManager, DepthManager, PopUpManager, StyleManager e SystemManager sono gestori.
- Schermate (mx.screens.*)
La categoria delle schermate comprende le classi ActionScript che consentono di controllare form e diapositive in Flash.

Per un elenco completo dei componenti, vedere *Guida di riferimento dei componenti*.

Informazioni sull'architettura dei componenti della versione 2

È possibile utilizzare la finestra di ispezione Proprietà o la finestra di ispezione dei componenti per cambiare i parametri dei componenti al fine di utilizzarne la funzionalità di base. Tuttavia, per avere un maggiore controllo sui componenti, usare le relative API e comprendere, per grandi linee, il modo in cui sono stati creati.

I componenti Flash sono stati creati con la versione 2 di Macromedia Component Architecture. I componenti della versione 2 sono supportati da Flash Player 6 (6.0.79.0) e versioni successive e da ActionScript 2.0. Questi componenti non sono sempre compatibili con quelli creati utilizzando l'architettura della versione 1 (tutti i componenti forniti prima di Flash MX 2004). Inoltre, i componenti originali della versione 1 non sono supportati da Flash Player 7. Per ulteriori informazioni, vedere [“Aggiornamento dei componenti della versione 1 all'architettura della versione 2” a pagina 67.](#)

NOTA

I componenti UI di Flash MX sono stati aggiornati per essere compatibili con Flash Player 7 o versioni successive. Questi componenti aggiornati sono sempre basati sull'architettura della versione 1. È possibile scaricarli dal sito di Macromedia Flash Exchange all'indirizzo www.macromedia.com/go/v1_components.

I componenti della versione 2 sono riportati nel pannello Componenti come simboli di clip compilato (SWC). I clip di questo tipo sono clip filmato di un componente il cui codice è stato compilato. I clip compilati non possono essere modificati ma è possibile cambiarne i parametri nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti, esattamente come per qualsiasi componente. Per ulteriori informazioni, vedere [“Informazioni su file SWC e clip compilati” a pagina 20.](#)

I componenti della versione 2 sono scritti in ActionScript 2.0. Ognuno di essi rappresenta una classe e ogni classe si trova in un pacchetto ActionScript. Ad esempio, un componente radio button è un'istanza della classe `RadioButton` il cui nome di pacchetto è `mx.controls`. Per ulteriori informazioni sui pacchetti, vedere [“Informazioni sui pacchetti” in *Apprendimento di ActionScript 2.0 in Flash*.](#)

La maggior parte dei componenti creati con la versione 2 di Macromedia Component Architecture è rappresentata da sottoclassi delle classi `UIObject` e `UIComponent` e ne eredita tutte le proprietà, i metodi e gli eventi. Molti componenti sono anche sottoclassi di altri componenti. Il percorso che indica l'ereditarietà di ciascun componente è indicato nella relativa voce della *Guida di riferimento dei componenti*.

NOTA

La gerarchia di classi è anch'essa disponibile come file `FlashPaper` nel percorso di installazione: `Flash 8\Samples and Tutorials\Samples\Components\arch_diagram.swf`.

Inoltre, tutti i componenti utilizzano lo stesso modello di evento, gli stessi stili CSS nonché gli stessi temi e meccanismi di skin incorporati. Per ulteriori informazioni su stili e skin, vedere il [Capitolo 5, “Personalizzazione dei componenti” a pagina 85](#). Per ulteriori informazioni sulla gestione degli eventi, consultare il [Capitolo 3, “Operazioni con i componenti” a pagina 53](#).

Per una descrizione dettagliata dell'architettura dei componenti della versione 2, consultare il [Capitolo 6, “Creazione dei componenti” a pagina 133](#).

Funzioni dei componenti della versione 2

In questa sezione vengono delineate le funzioni dei componenti della versione 2 (rispetto ai componenti della versione 1) dalla prospettiva di uno sviluppatore che utilizza i componenti per creare applicazioni Flash. Per informazioni dettagliate sulle differenze tra le architetture della versione 1 e della versione 2 per la creazione di componenti, consultare il [Capitolo 6, “Creazione dei componenti”](#) a pagina 133.

La **finestra di ispezione dei componenti** consente di cambiare i parametri dei componenti durante la fase di creazione sia in Macromedia Flash che in Macromedia Dreamweaver. (Vedere [“Impostazione dei parametri dei componenti”](#) a pagina 59.)

Il **modello di evento listener** consente ai listener di gestire gli eventi. (Vedere il [Capitolo 4, “Gestione degli eventi dei componenti”](#) a pagina 69.) La finestra di ispezione Proprietà non dispone di un parametro `clickHandler` come in Flash MX; per gestire gli eventi, è necessario scrivere codice ActionScript.

Le **proprietà degli skin** consentono di caricare singoli skin in fase di runtime, ad esempio le frecce su e giù o il segno di spunta di una casella di controllo. (Vedere [“Informazioni sull'associazione di skin ai componenti”](#) a pagina 101.)

Gli **stili CSS** consentono di creare un aspetto coerente e uniforme in tutte le applicazioni. (Vedere [“Uso degli stili per personalizzare il testo e il colore dei componenti”](#) a pagina 86.)

I **temi** consentono di trascinare un nuovo aspetto predefinito dalla libreria su una serie di componenti. (Vedere [“Informazioni sui temi”](#) a pagina 115.)

Halo è il tema predefinito utilizzato dai componenti della versione 2. (Vedere [“Informazioni sui temi”](#) a pagina 115.)

Le **classi di gestori** offrono un modo rapido di gestire l'attivazione e la profondità in un'applicazione. (Vedere [“Navigazione con attivazione personalizzata del componente”](#) a pagina 62 e [“Gestione della profondità dei componenti in un documento”](#) a pagina 64.)

Le **classi di base UIObject e UIComponent** forniscono i metodi, le proprietà e gli eventi principali per i componenti che estendono tali classi. (Vedere [“Classe UIComponent”](#) e [“Classe UIObject”](#) nella *Guida di riferimento dei componenti*.)

La distribuzione sotto forma di **pacchetto di file SWC** è facile e consente di nascondere il codice. Vedere il [Capitolo 6, “Creazione dei componenti”](#) a pagina 133.

L'**associazione di dati incorporata** è disponibile nella finestra di ispezione dei componenti. Per ulteriori informazioni, vedere [“Integrazione dei dati \(solo Flash Professional\)”](#) in *Uso di Flash*.

La **gerarchia di classi facile da estendere** utilizzando ActionScript 2.0 consente di creare spazi dei nomi univoci, importare agevolmente le classi necessarie e creare sottoclassi per estendere i componenti. Vedere [Capitolo 6, “Creazione dei componenti” a pagina 133](#) e la *Guida di riferimento di ActionScript 2.0*.

NOTA

Flash 8 include varie funzioni non supportate dai componenti della versione 2, inclusi 9-porzioni (o "scala 9"), FlashType e il caching bitmap.

Informazioni su file SWC e clip compilati

Un *clip compilato* è un pacchetto di simboli Flash precompilati e di codice ActionScript che consente di evitare la ricompilazione dei simboli e del codice che non si prevede di modificare. Un clip filmato può essere compilato anche in Flash e convertito in un clip compilato. Ad esempio, è possibile convertire in clip compilato un clip filmato contenente un'elevata quantità di codice ActionScript che non cambia di frequente. Il clip compilato si comporta esattamente come il clip filmato da cui è stato compilato, ma viene visualizzato e pubblicato molto più rapidamente rispetto al clip filmato normale. Il clip compilato non può essere modificato ma dispone di proprietà che vengono visualizzate nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti.

I componenti inclusi in Flash non sono file FLA ma clip compilati compressi in file clip compilati (SWC). SWC è il formato di file utilizzato da Macromedia per la distribuzione dei componenti e contiene un clip compilato, il file di classe ActionScript del componente e altri file che descrivono il componente. Per informazioni dettagliate sui file SWC, vedere [“Esportazione e distribuzione di un componente” a pagina 194](#).

Quando si inserisce un file SWC nella cartella First Run/Components, il componente appare nel pannello Componenti. Quando si aggiunge un componente allo stage dal pannello Componenti, alla libreria viene aggiunto un simbolo di clip compilato.

Per compilare un clip filmato:

- Fare clic con il pulsante destro del mouse (Windows) o premere Ctrl e fare clic (Macintosh) sul nome di un clip filmato nel pannello Libreria, quindi selezionare **Converti in clip compilato**.

Per esportare un file SWC:

- Selezionare il clip filmato nel pannello Libreria e fare clic con il pulsante destro del mouse (Windows) oppure fare clic tenendo premuto il tasto Ctrl (Macintosh), quindi selezionare Esporta file SWC.

NOTA

Flash Basic 8 e Flash Professional 8 continuano a supportare i componenti FLA.

Accessibilità e componenti

La necessità di rendere accessibile il contenuto del Web, ossia fruibile da portatori di handicap di vario tipo, è sempre più avvertita. È possibile rendere accessibile ai non vedenti il contenuto visivo delle applicazioni Flash mediante l'uso di software screen reader, che fornisce una descrizione vocale del contenuto dello schermo.

Quando si crea un componente, l'autore può creare codice ActionScript che consenta la comunicazione tra il componente e uno screen reader. Quando uno sviluppatore utilizza i componenti per creare un'applicazione in Flash, può servirsi del pannello Accessibilità per configurare ogni istanza del componente.

La maggior parte dei componenti creati da Macromedia è concepita per assicurare l'accessibilità. Per verificare se un componente è accessibile, vedere la relativa voce nella *Guida di riferimento dei componenti*. Quando si crea un'applicazione in Flash, è necessario aggiungere una riga di codice per ciascun componente

(`mx.accessibility.NomeComponenteAccImpl.enableAccessibility();`) e impostare i parametri di accessibilità nel pannello Accessibilità. Per i componenti, l'accessibilità funziona come nei clip filmato Flash.

Nella maggior parte dei componenti creati da Macromedia è possibile eseguire la navigazione anche tramite la tastiera. Ciascuna voce di componente descritta nella *Guida di riferimento dei componenti* indica se il componente può essere controllato o meno tramite la tastiera.

Creazione di un'applicazione con i componenti (solo Flash Professional)

I componenti sono elementi predefiniti che possono essere utilizzati per creare applicazioni Macromedia Flash. Includono controlli dell'interfaccia utente, meccanismi di connettività e accesso ai dati, oltre a elementi di tipo multimediale. I componenti semplificano il processo di creazione di un'applicazione Flash, in quanto forniscono gli elementi e il comportamento che, diversamente, sarebbe necessario creare da zero.

In questo capitolo viene presentata un'esercitazione che illustra come creare un'applicazione Flash utilizzando i componenti disponibili in Macromedia Flash Professional 8. Verrà illustrato inoltre come eseguire operazioni con i componenti nell'ambiente di creazione Flash e come renderli interattivi utilizzando codice ActionScript.

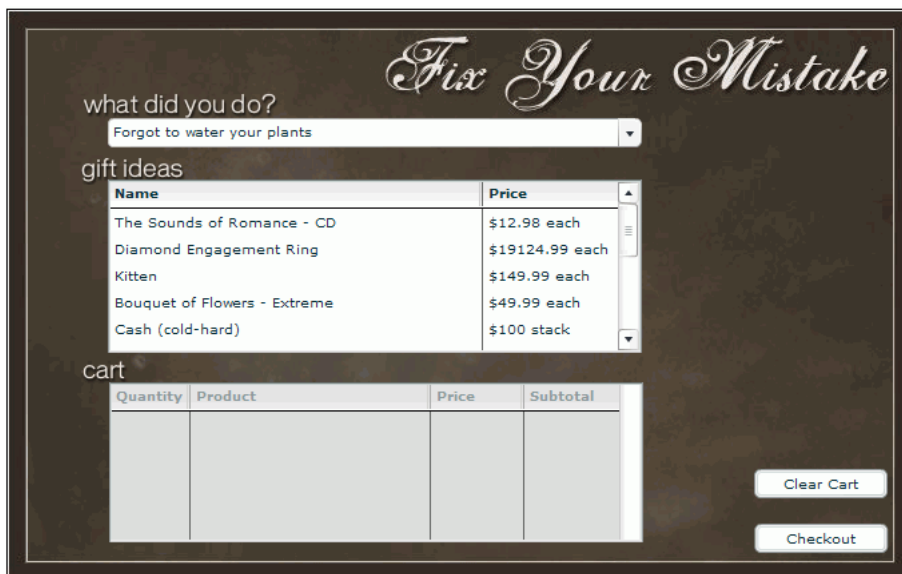
Informazioni sull'esercitazione Fix Your Mistake (Fatti perdonare)

In questa esercitazione viene illustrata la procedura per creare un'applicazione di base per gli acquisti in linea tramite il servizio di consegna regali "Fix Your Mistake". Il servizio consente agli utenti di scegliere un regalo appropriato per rimediare a una disattenzione compiuta nei confronti di un'altra persona. L'applicazione consente di filtrare un elenco di regali in base a scelte corrispondenti alla gravità della disattenzione. Dall'elenco l'utente può aggiungere articoli al carrello e quindi passare alla pagina Checkout e immettere le informazioni per la fatturazione, la spedizione e il pagamento con carta di credito.

Questo capitolo contiene le seguenti sezioni:

Informazioni sull'esercitazione Fix Your Mistake (Fatti perdonare)	23
Creazione della pagina principale	25
Associazione dei componenti dati per visualizzare le idee regalo	32
Visualizzazione dei dettagli dei regali.	37
Creazione della schermata Checkout	42
Prova dell'applicazione	51
Visualizzazione dell'applicazione completa	51

Per la creazione dell'interfaccia dell'applicazione vengono utilizzati i componenti ComboBox, DataGrid, TextArea, Button e altri. La pagina principale dell'interfaccia sarà analoga alla seguente:



Per la connessione dinamica a un servizio Web e per il recupero dell'elenco di disattenzioni (problems.xml) visualizzate nella casella combinata, viene utilizzata la classe ActionScript WebService. ActionScript viene impiegato inoltre per gestire le interazioni dell'utente con l'applicazione.

I componenti dati vengono utilizzati per connettere l'interfaccia a un'altra origine dati e il componente XMLConnector per connettere l'interfaccia a un file di dati XML (products.xml) contenente l'elenco di regali. Per filtrare i dati e presentarli nella griglia di dati, viene impiegato il componente DataSet.

Per questa esercitazione è necessaria una buona conoscenza dell'ambiente di creazione Flash e una discreta esperienza di ActionScript. Per quanto riguarda l'ambiente di creazione, è necessario conoscere l'uso dei pannelli, degli strumenti, della linea temporale e della libreria. In questa esercitazione viene fornito tutto il codice ActionScript necessario per creare l'applicazione di esempio. Per comprendere tuttavia i concetti relativi alla creazione degli script e creare applicazioni personalizzate, è necessario disporre di una discreta esperienza nella scrittura di codice ActionScript.

Per visualizzare una versione funzionante dell'applicazione finale, vedere [“Visualizzazione dell'applicazione completa”](#) a pagina 51.

Si tenga presente che l'applicazione di esempio ha uno scopo puramente dimostrativo e pertanto non è completa come potrebbe esserlo un'applicazione reale.

Creazione della pagina principale

Per creare la pagina principale dell'applicazione mediante l'aggiunta di componenti a una pagina iniziale schematica, seguire la procedura riportata sotto. Aggiungere quindi codice ActionScript per personalizzare i componenti, impostare le classi ActionScript che consentono di modificare i componenti dell'applicazione e accedere a un servizio Web per compilare la casella combinata con un elenco di disattenzioni. La casella combinata viene compilata dal codice se la proprietà `dataProvider` è impostata in modo da ricevere i risultati dal servizio Web.

1. Aprire il file `first_app_start.fla` disponibile in uno dei percorsi seguenti:
 - In Windows: *Unità di installazione*: \Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\ComponentsApplication
 - In Macintosh: *Macintosh HD*/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/ComponentsApplication

Il file contiene una pagina iniziale analoga alla seguente:



Il file `first_app_v3_start.fla` contiene tre livelli: il livello background con un'immagine di sfondo nera e titoli di testo, il livello text con le etichette di testo delle sezioni dell'applicazione e il livello labels con etichette sul primo fotogramma (Home) e sul decimo fotogramma (Checkout).

2. Selezionare File > Salva con nome. Rinominare il file e salvarlo sul disco rigido.
3. Nella linea temporale, selezionare il livello Labels e fare clic sul pulsante Inserisci livello per aggiungere un nuovo livello sopra di esso. Assegnare al nuovo livello il nome **Form**. Le istanze del componente vengono inserite in questo livello.
4. Accertarsi che il livello Form sia selezionato. Nel pannello Componenti (Finestra > Componenti), individuare il componente ComboBox nella struttura ad albero Interfaccia utente. Trascinare un'istanza di ComboBox nello stage posizionandola sotto il testo What Did You Do?. Nella finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà), immettere **problems_cb** come nome dell'istanza. Immettere **400** (pixel) come valore per la larghezza. Immettere **76,0** per l'asse *x* e **82,0** per l'asse *y*.

NOTA

Il simbolo del componente ComboBox viene aggiunto alla libreria (Finestra > Libreria). Quando si trascina un'istanza di componente nello stage, nella libreria viene aggiunto un simbolo di clip compilato relativo al componente. Analogamente a tutti i simboli usati in Flash, è possibile creare istanze aggiuntive del componente trascinando il simbolo della libreria nello stage.

5. Trascinare nello stage un'istanza del componente DataGrid dalla struttura ad albero Interfaccia utente del pannello Componenti e posizionarla sotto il testo Gift Ideas. Immettere **products_dg** come nome dell'istanza. Immettere **400** (pixel) come valore per la larghezza e **130** come valore per l'altezza. Immettere **76,0** per l'asse *x* e **128,0** per l'asse *y*.
6. Trascinare su un lato dello stage un'istanza del componente Dataset dalla struttura ad albero Dati del pannello Componenti. Il componente DataSet non viene visualizzato nell'applicazione in fase di runtime. L'icona DataSet è semplicemente un segnaposto che viene utilizzato nell'ambiente di creazione Flash. Immettere **products_ds** come nome dell'istanza.

Trascinare su un lato dello stage un'istanza del componente XMLConnector dalla struttura ad albero Dati del pannello Componenti. Analogamente al componente DataSet, il componente XMLConnector non viene visualizzato nell'applicazione in fase di runtime. Immettere **products_xmlcon** come nome dell'istanza. Fare clic sulla scheda Parametri nella finestra di ispezione Proprietà e immettere www.flash-mx.com/mm/firstapp/products.xml per la proprietà URL. Fare clic sul valore della proprietà di direzione per attivare la casella combinata, fare clic sulla freccia rivolta verso il basso e selezionare *receive* dall'elenco

NOTA

Per impostare i parametri dei componenti è inoltre possibile utilizzare la finestra di ispezione dei componenti (Finestra > Finestra di ispezione dei componenti) che funziona nello stesso modo della scheda Parametri della finestra di ispezione Proprietà.

L'URL specifica un file XML esterno contenente i dati sui prodotti visualizzati nella sezione Gift Ideas dell'applicazione. Più avanti nell'esercitazione viene utilizzata l'associazione dei dati per associare tra loro i componenti XMLConnector, DataSet e DataGrid; il componente DataSet filtra i dati del file XML esterno e il componente DataGrid li visualizza.

1. Trascinare nello stage un'istanza del componente Button dalla struttura ad albero Interfaccia utente del pannello Componenti posizionandola nell'angolo inferiore destro dello stage. Immettere **checkout_button** come nome dell'istanza. Fare clic sulla scheda Parametri e immettere **Checkout** (Pagamento) per la proprietà *label*. Per le coordinate *x* e *y* immettere rispettivamente **560,3** e **386,0**.

Importazione delle classi dei componenti

Ogni componente è associato a un file di classe `ActionScript` che ne definisce i metodi e le proprietà. In questa sezione dell'esercitazione viene aggiunto il codice `ActionScript` per importare le classi associate ai componenti dell'applicazione. Per alcuni componenti sono già state aggiunte istanze nello stage, mentre per altri il codice `ActionScript` viene aggiunto più avanti nell'esercitazione per creare le istanze in modo dinamico.

L'istruzione `import` crea un riferimento al nome della classe e facilita la scrittura del codice `ActionScript` per il componente. Consente inoltre di fare riferimento alla classe utilizzando il nome di classe anziché il nome completo, che comprende il nome del pacchetto. Ad esempio, dopo aver creato un riferimento al file della classe `ComboBox` con un'istruzione `import`, è possibile fare riferimento alle istanze di `ComboBox` con la sintassi `instanceName:ComboBox` anziché con la sintassi `instanceName:mx.controls.ComboBox`.

Un *pacchetto* rappresenta una directory che contiene i file di classe, inclusa a sua volta in una directory di percorso di classe prestabilita. È possibile utilizzare un carattere jolly per creare riferimenti a tutte le classi di un pacchetto: ad esempio, la sintassi `mx.controls.*` crea riferimenti a tutte le classi del pacchetto di controlli. Quando si utilizza un carattere jolly per creare un riferimento a un pacchetto, le classi non utilizzate vengono eliminate dall'applicazione al momento della compilazione, evitando di aumentare inutilmente le dimensioni.

Per l'applicazione in questa esercitazione, sono necessari i pacchetti e le classi singole seguenti:

Pacchetto di componenti controllo dell'interfaccia utente Questo pacchetto contiene le classi dei componenti controllo dell'interfaccia utente, quali ComboBox, DataGrid, Loader, TextInput, Label, NumericStepper, Button e CheckBox.

Pacchetto di componenti contenitore dell'interfaccia utente Questo pacchetto contiene le classi dei componenti contenitore dell'interfaccia utente, quali Accordion, ScrollPane e Window. Analogamente al pacchetto di controlli, è possibile utilizzare un carattere jolly per creare un riferimento al pacchetto.

Classe DataGridColumn Questa classe consente di aggiungere all'istanza DataGrid delle colonne e di controllarne l'aspetto.

Classe WebService Questa classe compila l'istanza ComboBox con un elenco di problemi o disattenzioni. Per questa classe occorre inoltre importare l'elemento WebServiceClasses dalla libreria comune Classes. Questo elemento contiene clip compilati (file SWC) necessari per compilare e generare il file SWF dell'applicazione.

Classe Cart Questa classe personalizzata viene fornita nell'esercitazione e definisce il funzionamento del carrello della spesa che verrà creato in seguito. Per esaminare il codice del file della classe Cart, aprire il file `cart.as` che si trova nella cartella `component_application` insieme ai file FLA e SWF dell'applicazione.

Per importare queste classi, creare un livello Actions e aggiungere il codice ActionScript al primo fotogramma della linea temporale principale. Tutto il codice che viene aggiunto all'applicazione nei punti successivi dell'esercitazione deve essere inserito nel livello Actions.

1. Per importare l'elemento WebServiceClasses dalla libreria Classes, selezionare Finestra > Librerie comuni > Classes.
2. Trascinare l'elemento WebServiceClasses dalla libreria Classes nella libreria dell'applicazione.

L'importazione di un elemento dalla libreria Classes è simile all'aggiunta di un componente alla libreria: i file SWC della classe vengono infatti aggiunti alla libreria. Per poter utilizzare la classe in un'applicazione, è necessario che i file SWC si trovino nella libreria.

3. Nella linea temporale, selezionare il livello Form e fare clic sul pulsante Inserisci livello. Assegnare al nuovo livello il nome **Actions**.
4. Con il livello Actions selezionato, selezionare il fotogramma 1 e premere F9 per aprire il pannello Azioni.
5. Nel pannello Azioni, immettere il codice seguente per creare una funzione `stop()` che impedisca la riproduzione ciclica dell'applicazione:

```
stop();
```

6. Mantenendo selezionato il fotogramma 1 del livello Actions, aggiungere il codice seguente nel pannello Azioni per importare le classi:

```
// Importa le classi necessarie
import mx.services.WebService;
import mx.controls.*;
import mx.containers.*;
import mx.controls.gridclasses.DataGridColumn;
// Importa la classe personalizzata Cart
import Cart;
```

Impostazione del tipo di dati delle istanze del componente

A questo punto, vengono assegnati i tipi di dati a ogni istanza del componente precedentemente trascinata nello stage.

In ActionScript 2.0 viene utilizzata la tipizzazione forte dei dati che prevede l'assegnazione del tipo di dati al momento della creazione di una variabile. Grazie alla tipizzazione forte dei dati, nel pannello Azioni vengono visualizzati suggerimenti sul codice della variabile.

- Nel pannello Azioni, aggiungere il codice seguente per assegnare il tipo di dati alle quattro istanze di componente già create.

```
/* Assegna il tipo di dati alle istanze nello stage; le altre istanze
   possono essere aggiunte in
   fase di runtime dalla classe Cart.*/
var problems_cb:ComboBox;
var products_dg:DataGrid;
var cart_dg:DataGrid;
var products_xmlcon:mx.data.components.XMLConnector;
```

NOTA

I nomi delle istanze specificati devono corrispondere ai nomi delle istanze assegnati quando i componenti sono stati trascinati nello stage.

Personalizzazione dell'aspetto dei componenti

Ogni componente dispone di proprietà di stile e di metodi che consentono di personalizzarne l'aspetto, modificando ad esempio il colore dell'evidenziazione, il carattere e le dimensioni del carattere. È possibile applicare gli stili singolarmente alle istanze di un componente oppure globalmente a tutte le istanze di componente di un'applicazione. In questa esercitazione, gli stili vengono applicati globalmente.

- Aggiungere il codice seguente per impostare gli stili:

```
// Definisce gli stili globali e le equazioni di andamento per ComboBox
problems_cb.
_global.style.setStyle("themeColor", "haloBlue");
_global.style.setStyle("fontFamily", "Verdana");
_global.style.setStyle("fontSize", 10);
_global.style.setStyle("openEasing",
    mx.transitions.easing.Bounce.easeOut);
```

Questo codice imposta il colore del tema, ovvero il colore di evidenziazione di un elemento selezionato, il carattere e le dimensioni del carattere per i componenti; imposta inoltre l'andamento per il componente ComboBox, ovvero il modo in cui viene visualizzato o nascosto l'elenco a discesa quando si fa clic sulla barra del titolo del componente ComboBox.

Visualizzazione delle disattenzioni nella casella combinata

In questa sezione viene aggiunto il codice per la connessione a un servizio Web contenente l'elenco delle disattenzioni, ad esempio Forgot to Water Your Plants (Ho dimenticato di bagnare le piante) e così via. Il file WSDL (Web Service Description Language) è disponibile nel sito www.flash-mx.com/mm/firstapp/problems.cfc?WSDL, dove è possibile visualizzarne la struttura.

Il codice ActionScript passa i risultati del servizio Web all'istanza ComboBox per la visualizzazione. Una funzione elenca le disattenzioni in ordine di gravità. Se il servizio Web non restituisce alcun risultato, ad esempio se il servizio non è attivo o non viene trovata la funzione, viene visualizzato un messaggio di errore nel pannello Output.

- Nel pannello Azioni, immettere il seguente codice:

```
/* Definisce il servizio Web utilizzato per recuperare una serie di problemi.
Questo servizio verrà associato all'istanza ComboBox problems_cb. */
var problemService:WebService = new WebService("http://www.flash-mx.com/mm/firstapp/problems.cfc?WSDL");
var myProblems:Object = problemService.getProblems();

/* Se si ottiene un risultato dal servizio Web, impostare il campo che verrà utilizzato per l'etichetta della colonna.
Imposta il fornitore di dati sui risultati restituiti dal servizio Web. */
myProblems.onResult = function(wsdResults:Array) {
    problems_cb.labelField = "name";
    problems_cb.dataProvider = wsdResults.sortOn("severity", Array.NUMERIC);
};

/* Se non è possibile connettersi al servizio Web remoto, visualizza i messaggi di errore nel pannello Output. */
myProblems.onFault = function(error:Object) {
    trace("error:");
    for (var prop in error) {
        trace("  "+prop+" -> "+error[prop]);
    }
};
```

SUGGERIMENTO

Premere Ctrl+S per salvare il documento, quindi premere Ctrl+Invio o scegliere Controllo > Prova filmato per provare l'applicazione. A questo punto la casella combinata dovrebbe essere compilata con un elenco di disattenzioni e si dovrebbe inoltre vedere la griglia di dati vuota creata per la sezione Gift Ideas (Idee regalo) insieme al pulsante Checkout.

Associazione dei componenti dati per visualizzare le idee regalo

All'inizio dell'esercitazione sono state aggiunte allo stage istanze dei componenti DataGrid, DataSet e XMLConnector. La proprietà URL dell'istanza XMLConnector, denominata `products_xmlcon`, è stata impostata sul percorso di un file XML contenente le informazioni sui prodotti per la sezione Gift Ideas dell'applicazione.

In questa fase, vengono utilizzate le funzioni di associazione dei dati dell'ambiente di creazione Flash per associare tra loro i componenti XMLConnector, DataSet e DataGrid, in modo che nell'applicazione vengano utilizzati i dati XML. Per informazioni generali sull'uso dell'associazione dei dati e di altre funzioni dell'architettura di integrazione dei dati Flash, consultare il Capitolo 16, "Integrazione dei dati (solo Flash Professional)" in *Uso di Flash*.

Quando si associano i componenti, il componente DataSet filtra l'elenco dei prodotti nel file XML in base alla gravità della disattenzione selezionata dall'utente nella sezione What Did You Do? (Cosa hai combinato?). L'elenco viene visualizzato dal componente DataGrid.

Uso di uno schema per descrivere l'origine dati XML

Quando si crea la connessione a un'origine dati XML esterna tramite il componente XMLConnector, è necessario specificare uno *schema* ovvero una rappresentazione schematica che descrive la struttura del documento XML. Lo schema indica al componente XMLConnector come interpretare l'origine dati XML. Il modo più semplice per specificare uno schema è importare una copia del file XML con il quale si desidera creare la connessione e utilizzare tale copia come schema.

1. Avviare il browser Web e visualizzare il sito www.flash-mx.com/mmm/firstapp/products.xml, ovvero il percorso impostato per il parametro URL di XMLConnector.
2. Selezionare File > Salva con nome.
3. Salvare `products.xml` nello stesso percorso del file FLA in uso.
4. Selezionare il fotogramma 1 nella linea temporale principale.
5. Accanto allo stage, selezionare l'istanza `products_xmlcon` (XMLConnector).
6. Nella finestra di ispezione dei componenti, fare clic sulla scheda Schema. Fare clic sul pulsante Importa sul lato destro della scheda Schema sopra il riquadro di scorrimento. Nella finestra di dialogo Apri, individuare il file `products.xml` importato nel punto 3 e fare clic su Apri. Lo schema del file `products.xml` viene visualizzato nel riquadro di scorrimento della scheda Schema.

Nel riquadro superiore della scheda Schema, selezionare l'elemento `image`. Nel riquadro inferiore, selezionare `data type` e modificare il valore da `<empty>` a `String`. Ripetere questo passaggio per l'elemento `description`.

Filtraggio dell'elenco delle idee regalo in base alla disattenzione

Per associare le istanze dei componenti `XMLConnector`, `DataSet` e `DataGrid` tra loro, è possibile utilizzare la scheda Associazioni della finestra di ispezione dei componenti.

Per informazioni sull'uso dell'associazione dei dati, vedere “Integrazione dei dati (solo Flash Professional)” in *Uso di Flash*.

1. Selezionare nello stage l'istanza `products_xmlcon` (`XMLConnector`) e fare clic sulla scheda Associazioni nella finestra di ispezione dei componenti.
2. Fare clic sul pulsante Aggiungi associazione.
3. Nella finestra di dialogo Aggiungi associazione, selezionare `results.products.product array` e fare clic su OK.
4. Nella scheda Associazioni, fare clic su `bound to` nel riquadro Attributi associazione, ovvero il riquadro inferiore contenente le coppie di nome attributo/valore.
5. Nella colonna Valore relativa all'elemento `bound to`, fare clic sull'icona della lente di ingrandimento per aprire la finestra di dialogo Collegato a.
6. Nella finestra di dialogo Collegato a, selezionare l'istanza `DataSet <products_ds>` nel riquadro Percorso componente. Selezionare `dataProvider:array` nel riquadro Ubicazione schema. Fare clic su OK.
7. Nella scheda Associazioni, fare clic su `direction` nel riquadro Attributi associazione. Nel menu a comparsa della colonna Valore, selezionare `out`.

Se si seleziona questa opzione, i dati passeranno dall'istanza `products_xmlcon` all'istanza `products_ds` anziché passare in entrambe le direzioni o dall'istanza `DataSet` all'istanza `XMLConnector`.

8. Nello stage, selezionare l'istanza `products_ds`. Nella scheda Associazioni della finestra di ispezione dei componenti, si noti che il fornitore di dati del componente viene visualizzato nel riquadro Elenco associazioni, nella parte superiore della scheda Associazioni. Nel riquadro Attributi associazione, il parametro `bound to` indica che l'istanza `products_ds` è associata all'istanza `products_xmlcon` e che la direzione di associazione è `in`.

Nella procedura che segue, l'istanza `DataSet` viene associata all'istanza `DataGrid` in modo da visualizzare i dati filtrati dal set nella griglia di dati.

9. Mantenere selezionata l'istanza `products_ds` e fare clic sul pulsante **Aggiungi associazione** nella scheda **Associazioni**.
10. Nella finestra di dialogo **Aggiungi associazione**, selezionare `dataProvider:array` e fare clic su **OK**.
11. Nella scheda **Associazioni**, verificare che in **Elenco associazioni** sia selezionato `dataProvider:array`.
12. Fare clic sull'elemento **bound to** nel riquadro **Attributi associazione**.
13. Nella colonna **Valore relativa** all'elemento **bound to**, fare clic sull'icona della lente di ingrandimento per aprire la finestra di dialogo **Collegato a**.
14. Nella finestra di dialogo **Collegato a**, selezionare l'istanza `products_dg` (**DataGrid**) nel riquadro **Percorso componente**. Selezionare `dataProvider:array` nel riquadro **Ubicazione schema**. Fare clic su **OK**.

Aggiunta di colonne alla sezione Gift Ideas

A questo punto, è possibile aggiungere le colonne alla griglia di dati della sezione **Gift Ideas** dell'applicazione per la visualizzazione delle informazioni e del prezzo dei prodotti.

- Selezionare il livello **Actions**. Nel pannello **Azioni**, aggiungere il codice seguente per creare, configurare e aggiungere all'istanza **DataGrid** una colonna **Name** (Nome) e una colonna **Price** (Prezzo):

```
// Definisce le colonne della griglia di dati e la relativa larghezza
// predefinita nell'istanza
// DataGrid products_dg.
var name_dgc:DataGridColumn = new DataGridColumn("name");
name_dgc.headerText = "Name";
name_dgc.width = 280;

// Aggiunge la colonna all'istanza DataGrid.
products_dg.addColumn(name_dgc);
var price_dgc:DataGridColumn = new DataGridColumn("price");
price_dgc.headerText = "Price";
price_dgc.width = 100;

// Definisce la funzione che verrà utilizzata per impostare l'etichetta
// della colonna
// in fase di runtime.
price_dgc.labelFunction = function(item:Object) {
    if (item != undefined) {
        return "$"+item.price+ " "+item.priceQualifier;
    }
};
products_dg.addColumn(price_dgc);
```

Attivazione di XMLConnector

A questo punto, è possibile aggiungere una riga di codice che indichi all'istanza XMLConnector di caricare, analizzare e associare i contenuti del file products.xml remoto. Questo file si trova nell'URL immesso per la proprietà URL dell'istanza XMLConnector creata in precedenza. Il file contiene le informazioni sui prodotti che appariranno nella sezione Gift Ideas dell'applicazione.

- Immettere il codice seguente nel pannello Azioni:

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

Aggiunta di un listener di eventi per filtrare l'elenco di idee regalo

In questa sezione viene aggiunto un listener di eventi per rilevare la selezione di una disattenzione nella sezione What Did You Do?, ovvero nell'istanza ComboBox problems_cb. Il listener include una funzione che filtra l'elenco Gift Ideas in base alla disattenzione scelta dall'utente. Se l'utente seleziona una disattenzione non grave, viene visualizzato un elenco di regali di valore modesto, ad esempio un CD o un mazzo di fiori, mentre se la disattenzione selezionata è più grave vengono proposti regali più costosi.

Per ulteriori informazioni sull'uso dei listener di eventi, vedere “Uso dei listener di eventi” in *Apprendimento di ActionScript 2.0 in Flash*.

- Nel pannello Azioni, immettere il seguente codice:

```
/* Definisce un listener per l'istanza ComboBox problems_cb.  
Questo listener filtra i prodotti in DataSet (e DataGrid).  
Il filtraggio è basato sulla gravità dell'elemento attualmente  
selezionato in 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) {  
        // Se la gravità dell'elemento corrente è maggiore o uguale  
        // all'elemento selezionato in ComboBox, restituisce true.  
        return (item.severity>=evt.target.selectedItem.severity);  
    };  
};  
  
// Aggiunge il listener a ComboBox.  
problems_cb.addEventListener("change", cbListener);
```

La reimpostazione della proprietà `filtered`, ovvero l'impostazione della proprietà su `false` e quindi su `true`, all'inizio della funzione `change()` assicura il funzionamento corretto della funzione nel caso in cui l'utente modifichi ripetutamente la selezione nella sezione What Did You Do?.

La funzione `filterFunc()` controlla se un determinato elemento della serie di regali rientra nella gravità selezionata dall'utente nella casella combinata. Se il regalo si trova nell'intervallo di gravità selezionato, viene visualizzato nell'istanza `DataGrid` che è associata all'istanza `DataSet`.

L'ultima riga di codice registra il listener nell'istanza `ComboBox problems_cb`.

Aggiunta del carrello

Il codice che viene aggiunto in seguito crea un'istanza della classe personalizzata `Cart` e quindi la inizializza.

- Nel pannello Azioni, immettere il seguente codice:

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

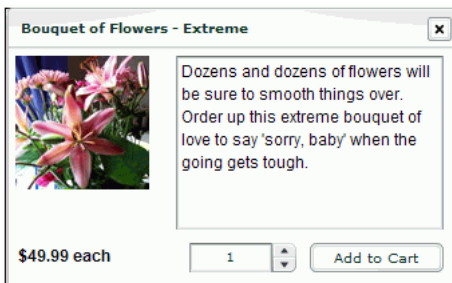
Questo codice utilizza il metodo `init()` della classe `Cart` per aggiungere un'istanza `DataGrid` allo stage, definire le colonne e posizionare l'istanza `DataGrid` nello stage. Il codice inoltre aggiunge e posiziona un'istanza del componente `Button` e aggiunge un gestore `Alert` per il pulsante. Per visualizzare il codice del metodo `init()` della classe `Cart`, aprire il file `Cart.as`.

SUGGERIMENTO

Premere Ctrl+S per salvare il documento, quindi premere Ctrl+Invio o scegliere Controllo-> Prova filmato per provare l'applicazione. Quando si seleziona una disattenzione nella casella combinata, nella griglia di dati creata per Gift Ideas dovrebbe essere visualizzato un sottoinsieme di regali corrispondente alla disattenzione selezionata.

Visualizzazione dei dettagli dei regali

Quando un utente fa clic su un prodotto nella sezione Gift Ideas, nell'applicazione viene visualizzata una finestra a comparsa contenente istanze di componenti che visualizzano le informazioni relative al prodotto, ad esempio un testo di descrizione, un'immagine e il prezzo. Per creare questa finestra a comparsa, viene creato un simbolo di clip filmato e vengono aggiunte istanze dei componenti Loader, TextArea, Label, NumericStepper e Button. La finestra di dettaglio del prodotto per Bouquet of Flowers Extreme (Mazzo di fiori d'emergenza) avrà il seguente aspetto:



Più avanti verrà aggiunto il codice ActionScript per la creazione dinamica di un'istanza di questo clip filmato per ogni prodotto. Queste istanze di clip filmato vengono visualizzate nel componente Window aggiunto in precedenza alla libreria. Le istanze dei componenti vengono compilate con gli elementi provenienti dal file XML esterno.

1. Trascinare nella libreria un'istanza del componente Window dalla struttura ad albero Interfaccia utente del pannello Componenti.
Il simbolo del componente Window viene aggiunto alla libreria. Più avanti nell'esercitazione verranno create istanze del componente Window tramite ActionScript.
2. Nel pannello Libreria (Finestra > Libreria), fare clic sul menu Opzioni nell'angolo destro della barra del titolo e selezionare Nuovo simbolo.
3. Nella finestra di dialogo Crea un nuovo simbolo, immettere **ProductForm** in Nome e selezionare Clip filmato in Tipo.
4. Fare clic sul pulsante Avanzato. Nell'area Concatenamento, selezionare Esporta per ActionScript, lasciare selezionata l'opzione Esporta nel primo fotogramma e fare clic su OK. Per il nuovo simbolo viene aperta una finestra del documento in modalità di modifica dei simboli.

Per i simboli di clip filmato contenuti nella libreria ma non nello stage, è necessario selezionare Esporta per ActionScript in modo da poterli modificare utilizzando ActionScript. (L'opzione Esporta nel primo fotogramma fa in modo che il clip filmato sia disponibile non appena viene caricato il primo fotogramma.) Più avanti nell'esercitazione verrà aggiunto il codice ActionScript per la generazione dinamica di un'istanza del clip filmato ogni volta che l'utente fa clic su un prodotto nella sezione Gift Ideas.

5. Nella linea temporale del nuovo simbolo, selezionare Livello 1 e rinominarlo **Components**.
6. Trascinare nello stage un'istanza del componente Loader dalla struttura ad albero Interfaccia utente del pannello Componenti Immettere 5 e 5 rispettivamente per le coordinate *x* e *y*. Immettere **image_ldr** come nome dell'istanza. Fare clic sulla scheda Parametri della finestra di ispezione Proprietà. Selezionare `false` per `autoLoad` e `false` per `scaleContent`.

L'istanza del componente Loader viene utilizzata per visualizzare un'immagine del prodotto. L'impostazione `false` per `autoLoad` specifica che l'immagine non viene caricata automaticamente. L'impostazione `false` per `scaleContent` specifica che l'immagine non viene adattata in scala automaticamente. Più avanti nell'esercitazione verrà aggiunto il codice per il caricamento dinamico dell'immagine corrispondente al prodotto che l'utente avrà selezionato nella sezione Gift Ideas.

7. Trascinare nello stage un'istanza del componente TextArea dalla struttura ad albero Interfaccia utente del pannello Componenti e posizionarla vicino al componente Loader. Immettere 125 e 5 rispettivamente per le coordinate *x* e *y*. Immettere **description_ta** come nome dell'istanza. Impostare la larghezza su 200 e l'altezza su 130. Fare clic sulla scheda Parametri della finestra di ispezione Proprietà. Per `editable`, selezionare `false`. Per `html`, selezionare `true`. Per `wordWrap`, selezionare `true`.

L'istanza del componente TextArea viene utilizzata per visualizzare una descrizione testuale del prodotto selezionato. Le impostazioni selezionate indicano che il testo non può essere modificato dall'utente, che può essere formattato con tag HTML e che le righe andranno a capo per adattare il testo alle dimensioni dell'area di testo.

8. Trascinare nello stage un'istanza del componente Label dalla struttura ad albero Interfaccia utente del pannello Componenti e posizionarla sotto al componente Loader. Impostare le coordinate *x* e *y* su 5 e 145. Immettere **price_lbl** come nome dell'istanza. Fare clic sulla scheda Parametri della finestra di ispezione Proprietà. Per `autoSize`, selezionare `left`. Per `html`, selezionare `true`.

L'istanza del componente Label visualizza il prezzo del prodotto e la quantità di prodotti indicati dal prezzo specificato, ad esempio `each` (cadauno) o `one dozen` (una dozzina).

9. Trascinare nello stage un'istanza del componente NumericStepper dalla struttura ad albero Interfaccia utente del pannello Componenti e posizionarla sotto al componente TextArea. Impostare le coordinate x e y su 135 e 145. Immettere **quantity_ns** come nome dell'istanza. Fare clic sulla scheda Parametri della finestra di ispezione Proprietà. Per `minimum`, immettere 1. L'impostazione di `minimum` su 1 indica che l'utente deve selezionare almeno uno dei prodotti per poter aggiungere l'articolo al carrello.
10. Trascinare nello stage un'istanza del componente Button dalla struttura ad albero Interfaccia utente del pannello Componenti e posizionarla accanto al componente NumericStepper. Impostare le coordinate x e y su 225 e 145. Immettere **addToCart_button** come nome dell'istanza. Fare clic sulla scheda Parametri della finestra di ispezione Proprietà. Per `label`, immettere **Add To Cart** (Aggiungi al carrello).

Aggiunta di un listener di eventi per attivare la visualizzazione dei dettagli dei regali

A questo punto, è possibile aggiungere un listener di eventi all'istanza `DataGrid products_dg` per visualizzare le informazioni su ogni prodotto. Quando l'utente fa clic su un prodotto nella sezione Gift Ideas (Idee regalo) viene visualizzata una finestra a comparsa contenente le informazioni sul prodotto.

- Nel pannello Azioni della linea temporale principale, immettere il codice seguente:

```
// Crea un listener per DataGrid per rilevare quando viene modificata la
// riga
// in DataGrid
var dgListener:Object = new Object();
dgListener.change = function(evt:Object) {
    // Quando la riga corrente viene modificata in DataGrid, viene aperta
    // una nuova
    // finestra a comparsa contenente i dettagli del prodotto.
    myWindow = mx.managers.PopUpManager.createPopUp(_root,
    mx.containers.Window, true, {title:evt.target.selectedItem.name,
    contentPath:"ProductForm", closeButton:true});
    // Imposta le dimensioni della finestra a comparsa.
    myWindow.setSize(340, 210);
    // Definisce un listener che chiude la finestra a comparsa quando
    // l'utente fa clic
    // sul pulsante di chiusura.
    var closeListener:Object = new Object();
    closeListener.click = function(evt) {
        evt.target.deletePopUp();
    };
    myWindow.addEventListener("click", closeListener);
};
products_dg.addEventListener("change", dgListener);
```

Questo codice crea un nuovo listener di eventi denominato `dgListener` e istanze del componente `Window` aggiunto precedentemente alla libreria. Il titolo della nuova finestra viene impostato sul nome del prodotto. Il percorso del contenuto della finestra viene impostato sul clip filmato `ProductForm`. Le dimensioni della finestra vengono impostate su 340 x 210 pixel.

Il codice aggiunge inoltre un pulsante di chiusura per consentire all'utente di chiudere la finestra dopo aver visualizzato le informazioni.

Aggiunta di codice al clip filmato `ProductForm`

A questo punto è necessario aggiungere il codice `ActionScript` al clip filmato `ProductForm` appena creato. Il codice `ActionScript` compila i componenti del clip filmato con le informazioni sul regalo selezionato e aggiunge un listener di eventi al pulsante `Add to Cart` (Aggiungi al carrello) che consente di aggiungere il prodotto selezionato al carrello.

Per ulteriori informazioni sull'uso dei listener di eventi, vedere “*Uso dei listener di eventi*” in *Uso di ActionScript in Flash*.

1. Nella linea temporale del clip filmato `ProductForm`, creare un nuovo livello e nominarlo **Actions**. Selezionare il primo fotogramma del livello **Actions**.
2. Nel pannello **Azioni**, immettere il seguente codice:

```
// Crea un oggetto che fa riferimento alla voce di prodotto selezionata
// in DataGrid.
var thisProduct:Object = this._parent._parent.products_dg.selectedItem;
// Compila le istanze TextArea description_ta e Label price_lbl con
// i dati del prodotto selezionato.
description_ta.text = thisProduct.description;
price_lbl.text = "<b>${thisProduct.price+"
               "+thisProduct.priceQualifier}</b>";
// Carica un'immagine del prodotto dalla directory dell'applicazione.
image_ldr.load(thisProduct.image);
```

NOTA

Il codice contiene commenti che ne descrivono lo scopo. È buona norma inserire commenti di questo tipo nel codice `ActionScript` che si sta creando, in modo che in seguito sia possibile comprenderne lo scopo per eventuali modifiche.

Innanzitutto, il codice definisce una variabile da utilizzare nel codice successivo per fare riferimento al prodotto selezionato. L'uso della variabile `thisProduct` consente di evitare di fare riferimento al prodotto specificato con il percorso

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


Quindi, il codice compila le istanze `TextArea` e `Label` utilizzando le proprietà `description`, `price` e `priceQualifier` dell'oggetto `thisProduct`. Queste proprietà corrispondono a elementi del file `products.xml` che sono stati collegati all'istanza `XMLConnector products_xml` con all'inizio dell'esercitazione. Più avanti nell'esercitazione, le istanze dei componenti `XMLConnector`, `DataSet` e `DataGrid` verranno associate l'una all'altra e gli elementi del file XML compileranno le altre due istanze di componente.

Infine, il codice utilizza la proprietà `image` dell'istanza dell'oggetto `thisProduct` per caricare un'immagine del prodotto nel componente `Loader`.

3. Aggiungere quindi un listener di eventi in modo che il prodotto venga aggiunto al carrello quando l'utente fa clic sul pulsante `Add to Cart` (Aggiungi al carrello). Più avanti nell'esercitazione, verrà aggiunto il codice `ActionScript` alla linea temporale principale dell'applicazione per creare un'istanza della classe `Cart`. Aggiungere il codice seguente:

```
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._parent.myCart;
    theCart.addProduct(tempObj.quantity, thisProduct);
};
addToCart_button.addEventListener("click", cartListener);
```

4. Fare clic sul pulsante `Controlla sintassi` (il segno di spunta blu sopra il riquadro `Script`) per verificare che il codice non contenga errori di sintassi.

Quando si aggiunge il codice a un'applicazione, è opportuno verificare la sintassi di frequente. Gli errori trovati nel codice vengono elencati nel pannello `Output`. Quando si esegue il controllo della sintassi, viene controllato solo lo script corrente, mentre eventuali altri script presenti nel file `FLA` non vengono verificati. Per ulteriori informazioni, consultare “Esecuzione del debug degli script” in *Apprendimento di ActionScript 2.0 in Flash*.

5. Fare clic sul pulsante freccia nella parte superiore sinistra della finestra del documento oppure selezionare Visualizza > Modifica documento per uscire dalla modalità di modifica dei simboli e tornare alla linea temporale principale.

SUGGERIMENTO

Premere Ctrl+S per salvare il documento, quindi premere Ctrl+Invio o scegliere Controllo > Prova filmato per provare l'applicazione. A questo punto, quando si fa clic sulla selezione di un regalo, viene visualizzata una finestra contenente l'immagine del regalo unitamente a una descrizione, al prezzo e a uno stepper numerico che consente di scegliere la quantità desiderata.

Creazione della schermata Checkout

Quando l'utente fa clic sul pulsante Checkout sulla schermata principale, viene visualizzata la schermata Checkout, che contiene i form nei quali l'utente può immettere le informazioni per la fatturazione, la spedizione e il pagamento con carta di credito. La schermata Checkout dovrebbe essere simile alla seguente:

checkout

Fix Your Mistake

1. Billing Information

First Name

Last Name

Country

Province/State

City

2. Shipping Information

3. Credit Card Information

Back

L'interfaccia per il pagamento è costituita da componenti posizionati su un fotogramma chiave del fotogramma 10 dell'applicazione e viene creata utilizzando il componente Accordion. Il componente Accordion è uno strumento di esplorazione contenente una sequenza di elementi secondari che vengono visualizzati uno alla volta. È opportuno aggiungere inoltre un'istanza del componente Button per creare un pulsante Back (Indietro) in modo da consentire all'utente di tornare alla schermata principale.

Più avanti nell'esercitazione, verranno creati clip filmato da utilizzare come elementi secondari nell'istanza Accordion per la visualizzazione dei riquadri Billing Information (Dati di fatturazione), Shipping Information (Dati di spedizione) e Credit Card Information (Dati carta di credito).

1. Nella linea temporale principale dell'applicazione, spostare l'indicatore di riproduzione sul fotogramma 10 (con etichetta Checkout). Accertarsi che sia selezionato il livello Form.
2. Inserire un fotogramma chiave vuoto nel fotogramma 10 del livello Form selezionando il fotogramma e quindi Inserisci > Linea temporale > Fotogramma chiave vuoto.
3. Selezionare il nuovo fotogramma chiave e trascinare nello stage un'istanza del componente Accordion dalla struttura ad albero Interfaccia utente del pannello Componenti. Nella finestra di ispezione Proprietà, immettere **checkout_acc** come nome di istanza. Impostare la larghezza su **300** pixel e l'altezza su **200** pixel.
4. Trascinare nell'angolo inferiore destro dello stage un'istanza del componente Button dalla struttura ad albero Interfaccia utente del pannello Componenti. Nella finestra di ispezione Proprietà, immettere **back_button** come nome di istanza. Fare clic sulla scheda Parametri e immettere **Back** (Indietro) per la proprietà `label`.

Informazioni sui riquadri per la fatturazione, la spedizione e il pagamento con carta di credito

I riquadri Billing Information (Dati di fatturazione), Shipping Information (Dati di spedizione) e Credit Card Information (Dati carta di credito) vengono creati con istanze di clip filmato visualizzate nell'istanza del componente Accordion. Ogni riquadro è costituito da due clip filmato annidati.

Il clip filmato principale contiene un componente ScrollPane, utilizzato per visualizzare il contenuto in un'area a scorrimento. Il clip filmato secondario contiene componenti Label e TextInput nei quali i clienti possono immettere dati personali, quali nome, indirizzo e così via. Il componente ScrollPane viene utilizzato per visualizzare il clip filmato secondario che consente all'utente di scorrere i campi contenenti le informazioni.

Creazione del riquadro Billing Information

Creare innanzitutto due clip filmato per visualizzare i campi del form Billing Information: un clip filmato principale con l'istanza del componente ScrollPane e un clip filmato secondario con le istanze dei componenti Label e TextArea.

1. Nel pannello Libreria (Finestra > Libreria), fare clic sul menu Opzioni nell'angolo destro della barra del titolo e selezionare Nuovo simbolo.
2. Nella finestra di dialogo Crea un nuovo simbolo, immettere **checkout1_mc** in Nome e selezionare Clip filmato in Tipo.
3. Fare clic sul pulsante Avanzato. Nell'area Concatenamento, selezionare Esporta per ActionScript, lasciare selezionata l'opzione Esporta nel primo fotogramma e fare clic su OK.

Per il nuovo simbolo viene aperta una finestra del documento in modalità di modifica dei simboli.

4. Trascinare un'istanza del componente ScrollPane nello stage.
5. Nella finestra di ispezione Proprietà, immettere **checkout1_sp** come nome di istanza. Impostare i valori L e A su **300** e **135**. Impostare le coordinate x e y su **0** e **0**.
6. Fare clic sulla scheda Parametri. Impostare la proprietà `contentPath` su **checkout1_sub_mc**.

Il clip filmato checkout1_sub_mc verrà visualizzato all'interno del riquadro di scorrimento e conterrà i componenti Label e TextInput. Questo clip filmato verrà ora creato.

7. Dal menu Opzioni del pannello Libreria, selezionare Nuovo simbolo.
8. Nella finestra di dialogo Crea un nuovo simbolo, immettere **checkout1_sub_mc** in Nome e selezionare Clip filmato in Tipo.
9. Fare clic sul pulsante Avanzato. Nell'area Concatenamento, selezionare Esporta per ActionScript, lasciare selezionata l'opzione Esporta nel primo fotogramma e fare clic su OK.

Per il nuovo simbolo viene aperta una finestra del documento in modalità di modifica dei simboli.

10. Trascinare sei istanze del componente Label nello stage. In alternativa, è possibile trascinare una sola istanza nello stage e quindi trascinarla a sua volta nello stage tenendo premuto il tasto Ctrl (Windows) o il tasto Opzione (Macintosh) per creare le copie. Assegnare un nome e una posizione alle istanze nel modo indicato di seguito:

- Per la prima istanza, immettere **firstname_lbl** come nome dell'istanza e impostare le coordinate x e y su **5** e **5**. Fare clic sulla scheda Parametri e immettere **First Name** (Nome) per la proprietà `text`.

- Per la seconda istanza, immettere **lastname_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 35. Fare clic sulla scheda Parametri e immettere **Last Name** (Cognome) per la proprietà `text`.
 - Per la terza istanza, immettere **country_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 65. Fare clic sulla scheda Parametri e immettere **Country** (Paese) per la proprietà `text`.
 - Per la quarta istanza, immettere **province_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 95. Fare clic sulla scheda Parametri e immettere **Province/State** (Provincia/Stato) per la proprietà `text`.
 - Per la quinta istanza, immettere **city_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 125. Fare clic sulla scheda Parametri e immettere **City** (Città) per la proprietà `text`.
 - Per la sesta istanza, immettere **postal_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 155. Fare clic sulla scheda Parametri e immettere **Postal/Zip code** (CAP) per la proprietà `text`.
11. Trascinare sei istanze del componente TextInput nello stage. Posizionare un'istanza TextInput immediatamente a destra di ogni istanza Label. Ad esempio, le coordinate x e y della prima istanza TextInput dovrebbero essere 105, 5. Assegnare un nome alle istanze TextInput nel modo indicato di seguito:
- Prima istanza: **billingFirstName_ti**.
 - Seconda istanza: **billingLastName_ti**.
 - Terza istanza: **billingCountry_ti**.
 - Quarta istanza: **billingProvince_ti**.
 - Quinta istanza: **billingCity_ti**.
 - Sesta istanza: **billingPostal_ti**.
- In alcuni casi, è possibile ritagliare il contenuto di un riquadro a scorrimento che si trova troppo vicino al bordo del riquadro. La procedura che segue consente di aggiungere un rettangolo bianco al clip filmato `checkout1_sub_mc`, in modo da visualizzare correttamente le istanze Label e TextInput.
12. Nella linea temporale, fare clic sul pulsante Inserisci livello. Trascinare il nuovo livello sotto il livello esistente. Il livello contenente il rettangolo deve trovarsi in basso in modo che il rettangolo non interferisca con la visualizzazione del componente.
13. Selezionare il fotogramma 1 del nuovo livello.
14. Nel pannello Strumenti, selezionare lo strumento Rettangolo. Impostare nessun colore per il tratto e il bianco come colore di riempimento.

Fare clic sul controllo Colore tratto del pannello Strumenti e quindi sul pulsante Nessuno (il campione bianco barrato da una riga rossa diagonale). Quindi, fare clic sul controllo Colore riempimento e quindi sul campione di colore del bianco.

15. Trascinare il cursore per creare un rettangolo che si estenda oltre i bordi inferiore e destro delle istanze Label e TextInput.

Creazione del riquadro Shipping Information

I clip filmato del riquadro Shipping Information sono molto simili a quelli del riquadro Billing Information. Verrà aggiunto inoltre un componente CheckBox per consentire agli utenti di compilare i campi del form Shipping Information con gli stessi dati immessi nel riquadro Billing Information.

1. Per creare i clip filmato per il riquadro Credit Card Information, seguire le istruzioni precedenti (vedere [“Creazione del riquadro Billing Information” a pagina 44](#)). Si notino le seguenti differenze di denominazione:
 - Per il primo clip filmato, immettere **checkout2_mc** come nome del simbolo e **checkout2_sp** come nome dell'istanza. Nella scheda Parametri della finestra di ispezione Proprietà, impostare la proprietà `contentPath` su **checkout2_sub_mc**.
 - Per il secondo clip filmato, immettere **checkout2_sub_mc** come nome del simbolo.
 - Per le istanze TextInput, sostituire "billing" con "shipping" nel nome delle istanze.
2. Con il clip filmato **checkout2_sub_mc** aperto in modalità di modifica dei simboli, trascinare un'istanza del componente CheckBox nello stage e posizionarla appena sopra la prima istanza Label.

Assicurarsi di posizionare questa istanza sul Livello 1 insieme alle altre istanze di componente.
3. Nella finestra di ispezione Proprietà, immettere **sameAsBilling_ch** come nome di istanza.
4. Fare clic sulla scheda Parametri. Impostare la proprietà `label` su **Same As Billing Info**.

Creazione del riquadro Credit Card Information

Anche i clip filmato del riquadro Credit Card Information sono simili a quelli dei riquadri Billing Information e Shipping Information. Tuttavia, il clip filmato annidato del riquadro Credit Card Information dispone di campi leggermente diversi rispetto agli altri due riquadri, ovvero di campi per il numero della carta di credito e altri dati relativi alla carta.

1. Per creare i clip filmato del riquadro Credit Card Information, eseguire i punti 1-9 delle istruzioni relative al riquadro Billing Information indicate in [“Creazione del riquadro Billing Information” a pagina 44](#). Si notino le seguenti differenze di denominazione:

- Per il primo clip filmato, immettere **checkout3_mc** come nome del simbolo e **checkout3_sp** come nome dell'istanza. Nella scheda Parametri della finestra di ispezione Proprietà, impostare la proprietà `contentPath` su **checkout3_sub_mc**.
 - Per il secondo clip filmato, immettere **checkout3_sub_mc** come nome del simbolo.
2. Trascinare quattro istanze del componente Label nello stage. Assegnare un nome e una posizione alle istanze nel modo indicato di seguito:
 - Per la prima istanza, immettere **ccName_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 5. Fare clic sulla scheda Parametri e immettere **Name on Card** (Titolare della carta) per la proprietà `text`.
 - Per la seconda istanza, immettere **ccType_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 35. Fare clic sulla scheda Parametri e immettere **Card Type** (Tipo carta) per la proprietà `text`.
 - Per la terza istanza, immettere **ccNumber_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 65. Fare clic sulla scheda Parametri e immettere **Card Number** (Numero carta) per la proprietà `text`.
 - Per la quarta istanza, immettere **ccExp_lbl** come nome dell'istanza e impostare le coordinate x e y su 5 e 95. Fare clic sulla scheda Parametri e immettere **Expiration** (Scadenza) per la proprietà `text`.
 3. Trascinare nello stage un'istanza del componente TextInput e posizionarla a destra dell'istanza **ccName_lbl**. Assegnare alla nuova istanza il nome **ccName_ti**. Impostare le coordinate x e y su 105 e 5. Impostare la larghezza su 140.
 4. Trascinare un'istanza del componente ComboBox nello stage e posizionarla a destra dell'istanza **ccType_lbl**. Assegnare alla nuova istanza il nome **ccType_cb**. Impostare le coordinate x e y su 105 e 35. Impostare la larghezza su 140.
 5. Trascinare un'altra istanza del componente TextInput nello stage e posizionarla a destra dell'istanza **ccNumber_lbl**. Assegnare alla nuova istanza il nome **ccNumber_ti**. Impostare le coordinate x e y su 105 e 65. Impostare la larghezza su 140.
 6. Trascinare due istanze del componente ComboBox nello stage. Posizionare un'istanza a destra dell'istanza **ccExp_lbl** e l'altra a destra dell'istanza aggiunta. Assegnare alla prima nuova istanza il nome **ccMonth_cb**. Impostare la larghezza su 60 e le coordinate x e y su 175 e 95. Denominare la seconda istanza **ccYear_cb**. *Impostare la larghezza su 70 e le coordinate x e y su 175 e 95.
 7. Trascinare un'istanza del componente Button nello stage e posizionarla nella parte inferiore del form sotto l'istanza **ccMonth_cb**. Assegnare alla nuova istanza il nome **checkout_button**. Impostare le coordinate x e y su 125 e 135. Nella scheda Parametri della finestra di ispezione Proprietà, impostare la proprietà `label` su **Checkout**.

8. Per aggiungere un rettangolo nella parte inferiore del form, eseguire i punti 14-15 delle istruzioni relative al riquadro Billing Information indicate in [“Creazione del riquadro Billing Information” a pagina 44.](#)

Aggiunta di un listener di eventi al pulsante Checkout

È quindi possibile aggiungere il codice per visualizzare la schermata Checkout (Pagamento) quando l'utente fa clic sul pulsante Checkout.

- Nel pannello Azioni della linea temporale principale, aggiungere il codice seguente:

```
// Quando si fa clic sul pulsante Checkout, passa all'etichetta del
  fotogramma "checkout"
var checkoutBtnListener:Object = new Object();
checkoutBtnListener.click = function(evt:Object) {
    evt.target._parent.gotoAndStop("checkout");
};
checkout_button.addEventListener("click", checkoutBtnListener);
```

Questo codice indica che quando l'utente fa clic sul pulsante Checkout, l'indicatore di riproduzione si sposta sull'etichetta Checkout nella linea temporale.

Aggiunta di codice per la schermata Checkout

È possibile ora aggiungere il codice alla schermata Checkout dell'applicazione nel fotogramma 10 della linea temporale principale. Questo codice elabora i dati immessi dagli utenti nei riquadri Billing Information, Shipping Information e Credit Card Information creati in precedenza utilizzando il componente Accordion e altri componenti.

1. Nella linea temporale, selezionare il fotogramma 10 nel livello Actions e inserire un fotogramma vuoto selezionando Inserisci > Linea temporale > Fotogramma chiave vuoto.
2. Aprire il pannello Azioni (F9).
3. Nel pannello Azioni, immettere il seguente codice:

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

// Definisce il componente Accordion nello stage.
var checkout_acc:Accordion;
```

4. Aggiungere il primo elemento secondario dell'istanza del componente Accordion per accettare i dati di fatturazione immessi dall'utente. Aggiungere il codice seguente:

```
// Definisce l'elemento secondario del componente Accordion.
var child1 = checkout_acc.createChild("checkout1_mc", "child1_mc",
    {label:"1. Billing Information"});
var thisChild1 = child1.checkout1_sp.spContentHolder;
```


La prima riga chiama il metodo `createChild()` del componente `Accordion` e crea un'istanza del simbolo di clip filmato `checkout1_mc` creato in precedenza con il nome di istanza `child1_mc` e l'etichetta "1. Billing Information". La seconda riga di codice crea un collegamento a un'istanza del componente `ScrollPane` incorporata.

5. Creare il secondo elemento secondario dell'istanza `Accordion` per accettare i dati di fatturazione:

```
/* Aggiunge il secondo elemento secondario ad Accordion.  
Aggiunge un listener di eventi per CheckBox sameAsBilling_ch.  
Copia i valori del form dal primo elemento secondario nel secondo  
elemento secondario. */  
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);
```

Le prime due righe di codice sono simili al codice per la creazione dell'elemento secondario `Billing Information`: viene creata un'istanza del simbolo di clip filmato `checkout2_mc` con il nome di istanza `child2_mc` e l'etichetta "2. Shipping Information". La seconda riga di codice crea un collegamento a un'istanza del componente `ScrollPane` incorporata.

A partire dalla terza riga di codice, viene aggiunto un listener di eventi all'istanza `CheckBox`. Se l'utente fa clic sulla casella di controllo, come dati di fatturazione vengono utilizzate le informazioni immesse dall'utente nel riquadro `Billing Information`.

6. Creare quindi un terzo elemento secondario per l'istanza `Accordion` per le informazioni relative alla carta di credito:

```
// Definisce il terzo elemento secondario di Accordion.  
var child3 = checkout_acc.createChild("checkout3_mc", "child3_mc",  
    {label:"3. Credit Card Information"});  
var thisChild3 = child3.checkout3_sp.spContentHolder;
```

7. Aggiungere questo codice per creare istanze ComboBox per il mese, l'anno e il tipo di carta di credito e compilare ogni istanza con un array definito staticamente:

```
/* Imposta i valori delle tre istanze ComboBox nello stage:  
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. Infine, aggiungere il codice seguente per aggiungere listener di eventi ai pulsanti Checkout (Pagamento) e Back (Indietro). Quando l'utente fa clic sul pulsante Checkout, l'oggetto listener copia i campi del form dai riquadri Billing Information, Shipping Information e Credit Card Information in un oggetto LoadVars che viene inviato al server. La classe LoadVars consente di inviare tutte le variabili di un oggetto a un URL specifico. Quando l'utente fa clic sul pulsante Back (Indietro), viene visualizzata di nuovo la schermata principale.

```
/* Crea un listener per l'istanza Button checkout_button.  
Questo listener invia tutte le variabili del form al server quando  
l'utente fa clic sul pulsante Checkout (Pagamento). */  
var checkoutListener:Object = new Object();  
checkoutListener.click = function(evt:Object){  
    evt.target.enabled = false;  
    /* Crea due istanze dell'oggetto LoadVars alle quali vengono inviate  
    le variabili  
    e ricevono i risultati dal server remoto. */  
    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;
```

```

/* Invia le variabili provenienti da checkout_lv LoadVars allo script
remoto sul server.
Salva i risultati nell'istanza 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);

```

Prova dell'applicazione

La creazione dell'applicazione è terminata. Premere Ctrl+S per salvare il documento, quindi premere Ctrl+Invio o scegliere Controllo > Prova filmato per provare l'applicazione.

Visualizzazione dell'applicazione completa

Qualora l'esercitazione non sia stata completata correttamente, è tuttavia possibile visualizzare una versione funzionante dell'applicazione finale. È possibile trovare questo file di inizio Flash (FLA), *first_app_start.fla*, nella cartella *Samples* sul disco rigido:

- In Windows: *Unità di avvio:\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/ComponentsApplication*.

Per visualizzare il file FLA dell'applicazione, aprire il file *first_app.fla* contenuto nella cartella *components_application*.

È possibile confrontare questi file con quelli creati durante l'esercitazione per individuare più facilmente eventuali errori.

Nella libreria vengono elencati tutti i componenti utilizzati nell'applicazione insieme ai file grafici e ad altre risorse utilizzate per creare l'applicazione. Alcuni componenti vengono visualizzati nello stage sotto forma di istanze; altri sono inclusi nel codice ActionScript e non vengono visualizzati fino alla fase di runtime.

Questo capitolo spiega come aggiungere i componenti a un documento e impostarne le proprietà mediante vari file di Macromedia Flash (FLA) e file di classe ActionScript. Verranno inoltre illustrati alcuni argomenti avanzati, quali l'uso dei suggerimenti sul codice, la navigazione con attivazione personalizzata, la gestione della profondità dei componenti e l'aggiornamento dei componenti della versione 1 alla versione 2 di Macromedia Component Architecture.

I file utilizzati in questo capitolo sono TipCalculator.fla e TipCalculator.swf, installati nei seguenti percorsi del disco rigido:

- (Windows) Programmi\Macromedia\Flesh 8\Samples and Tutorials\Samples\Components\TipCalculator
- (Macintosh) Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/TipCalculator

In questo capitolo vengono illustrati i seguenti argomenti:

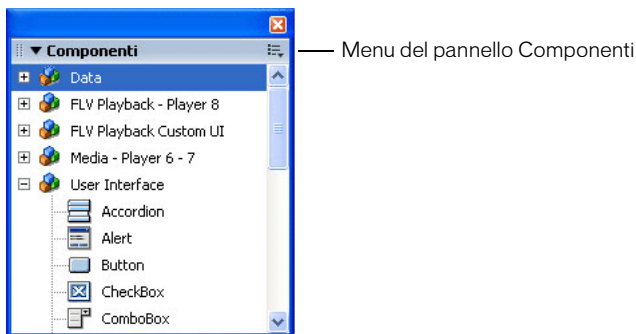
Pannello Componenti	54
Aggiunta di componenti ai documenti Flash	54
Componenti del pannello Libreria	58
Impostazione dei parametri dei componenti	59
Ridimensionamento dei componenti	61
Eliminazione dei componenti dai documenti Flash	62
Uso dei suggerimenti sul codice	62
Navigazione con attivazione personalizzata del componente	62
Gestione della profondità dei componenti in un documento	64
Componenti in Anteprima dal vivo	64
Uso di un precaricatore con i componenti	65
Informazioni sul caricamento dei componenti	66
Aggiornamento dei componenti della versione 1 all'architettura della versione 2	67

Pannello Componenti

Il pannello Componenti contiene tutti i componenti della directory Configuration/Components a livello utente. Per ulteriori informazioni su questa directory, vedere [“Percorso di memorizzazione dei file dei componenti”](#) a pagina 14.

Per visualizzare il pannello Componenti:

- Selezionare Finestra > Componenti.



Per visualizzare i componenti installati dopo l'avvio di Flash:

1. Selezionare Finestra > Componenti.
2. Selezionare Ricarica dal menu a comparsa del pannello Componenti.

Aggiunta di componenti ai documenti Flash

Quando si trascina nello stage un componente dal pannello Componenti, nel pannello Libreria viene aggiunto un simbolo di clip compilato (SWC). Una volta aggiunto un simbolo SWC alla libreria, è possibile trascinarne più istanze nello stage. Questo componente può inoltre essere aggiunto a un documento in fase di runtime mediante il `UIObject.createClassObject()` metodo ActionScript.

NOTA

I componenti Menu e Alert sono due eccezioni e non possono essere trasformati in istanze utilizzando `UIObject.createClassObject()`. Essi utilizzano invece il metodo `show()`.

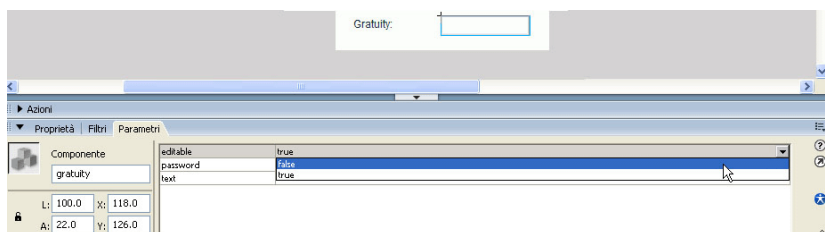
Aggiunta di componenti durante la fase di creazione

È possibile aggiungere un componente in un documento mediante il pannello Componenti e quindi aggiungerne ulteriori istanze trascinando nello stage il componente dal pannello Libreria. Le proprietà delle istanze aggiuntive possono essere impostate nella scheda Parametri della finestra di ispezione Proprietà o nella finestra di ispezione dei componenti.

Per aggiungere un componente a un documento Flash mediante il pannello Componenti:

1. Selezionare Finestra > Componenti.
2. Effettuare una delle seguenti operazioni:
 - Trascinare un componente nello stage dal pannello Componenti.
 - Fare doppio clic su un componente nel pannello Componenti.
3. Se il componente è un file FLA (tutti i componenti della versione 2 installati sono file SWC) e sono stati modificati gli skin per un'altra istanza dello stesso componente o per un componente che condivide gli skin con il componente che si sta aggiungendo, effettuare una delle seguenti operazioni:
 - Selezionare Non sostituire elementi esistenti per mantenere gli elementi skin modificati e applicarli al nuovo componente.
 - Selezionare Sostituisci gli elementi esistenti per sostituire tutti gli elementi skin con quelli predefiniti. Il nuovo componente e tutte le versioni precedenti del componente, o dei componenti che condividono gli stessi elementi skin, useranno gli elementi skin predefiniti.
4. Selezionare il componente sullo stage.
5. Selezionare Finestra > Proprietà > Proprietà.
6. Nella finestra di ispezione Proprietà, immettere un nome per l'istanza del componente.
7. Fare clic sulla scheda Parametri e specificare i parametri dell'istanza.

Nell'illustrazione seguente viene mostrata la finestra di ispezione Proprietà del componente TextInput contenuto nel file di esempio TipCalculator.fla installato in Flash 8/Samples and Tutorials/Samples/Components/TipCalculator.



Per ulteriori informazioni, vedere [“Impostazione dei parametri dei componenti” a pagina 59](#).

8. Ridimensionare il componente nel modo desiderato modificando i valori per larghezza e altezza.

Per ulteriori informazioni sul ridimensionamento di determinati tipi di componenti, vedere le voci relative ai singoli componenti nella *Guida di riferimento dei componenti*.

9. Se si desidera modificare il colore e la formattazione del testo di un componente, effettuare una o più delle operazioni seguenti:
 - Impostare o modificare il valore di una determinata proprietà dello stile per l'istanza di un componente utilizzando il metodo `setStyle()`, che è disponibile per tutti i componenti. Per ulteriori informazioni, vedere `UIObject.setStyle()` [a pagina 1432](#).
 - Modificare più proprietà nella dichiarazione di stile globale assegnata a tutti i componenti della versione 2.
 - Creare una dichiarazione di stile personalizzata per istanze specifiche del componente. Per ulteriori informazioni, vedere [“Uso degli stili per personalizzare il testo e il colore dei componenti” a pagina 86](#).
10. Se si desidera personalizzare l'aspetto del componente, effettuare una delle seguenti operazioni:
 - Applicare un tema (vedere [“Informazioni sui temi” a pagina 115](#)).
 - Modificare gli skin del componente (vedere [“Informazioni sull'associazione di skin ai componenti” a pagina 101](#)).

Aggiunta di componenti in fase di runtime mediante ActionScript

le istruzioni contenute in questa sezione presuppongono una conoscenza intermedia o avanzata di ActionScript.

Per aggiungere a un'applicazione Flash dei componenti in modo dinamico, utilizzare il metodo `createClassObject()`, ereditato dalla maggior parte dei componenti dalla classe `UIObject`. Ad esempio, è possibile aggiungere componenti che creano un layout di pagina in base alle preferenze impostate dall'utente, come nella home page di molti portali Web.

I componenti della versione 2 installati con Flash si trovano nelle directory di pacchetto. Per ulteriori informazioni, vedere “Informazioni sui pacchetti” in *Apprendimento di ActionScript 2.0 in Flash*. Se si aggiunge un componente nello stage durante la fase di creazione, è possibile fare riferimento ad esso semplicemente utilizzandone il nome di istanza, ad esempio `myButton`. Tuttavia, se si aggiunge un componente a un'applicazione in fase di runtime mediante ActionScript, specificarne il nome di classe completo, ad esempio `mx.controls.Button`, oppure importare il pacchetto utilizzando l'istruzione `import`.

Ad esempio, per scrivere il codice ActionScript che si riferisce a un componente `Alert`, è possibile utilizzare l'istruzione `import` per fare riferimento alla classe, nel modo seguente:

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

In alternativa, è possibile utilizzare il percorso di pacchetto completo, nel modo seguente:

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

Per ulteriori informazioni, vedere “Informazioni sull'importazione di file di classe” in *Apprendimento di ActionScript 2.0 in Flash*.

È possibile usare i metodi di ActionScript per impostare parametri aggiuntivi per i componenti aggiunti in modo dinamico. Per ulteriori informazioni, vedere la *Guida di riferimento dei componenti*.

NOTA

Per aggiungere un componente a un documento in fase di runtime, è necessario che il componente si trovi nella libreria quando viene compilato il file SWF. Per aggiungere un componente alla libreria, trascinare l'icona del componente dal pannello Componenti alla libreria. Se inoltre si carica un clip filmato contenente un componente trasformato dinamicamente, tramite ActionScript, in istanza di un altro clip filmato, è necessario che al momento della compilazione del file SWF il componente del clip filmato principale sia presente nella libreria.

Per aggiungere un componente al documento Flash mediante ActionScript:

1. Trascinare un componente dal pannello Componenti nella libreria del documento corrente.

NOTA

L'impostazione predefinita dei componenti è Esporta nel primo fotogramma (fare clic con il pulsante destro del mouse per Windows, o tenendo premuto Control per Macintosh, e selezionare l'opzione di menu Concatenamento per visualizzare l'impostazione Esporta nel primo fotogramma). Se si desidera utilizzare un precaricatore per un'applicazione che include componenti, è necessario modificare il fotogramma di esportazione. Vedere “[Uso di un precaricatore con i componenti](#)” a pagina 65 per istruzioni.

2. Nella linea temporale, selezionare il fotogramma in cui si desidera aggiungere il componente.

3. Aprire il pannello Azioni nel caso in cui non sia già visualizzato.
4. Chiamare il metodo `createClassObject()` per creare l'istanza del componente in fase di runtime.

Questo metodo può essere chiamato autonomamente o da qualsiasi istanza del componente. Il metodo `createClassObject()` accetta come parametri il nome di classe del componente, il nome della nuova istanza, la profondità e un oggetto di inizializzazione facoltativo che è possibile utilizzare per impostare le proprietà in fase di runtime.

Nel parametro del nome di classe è possibile specificare il pacchetto della classe, come illustrato nell'esempio seguente:

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

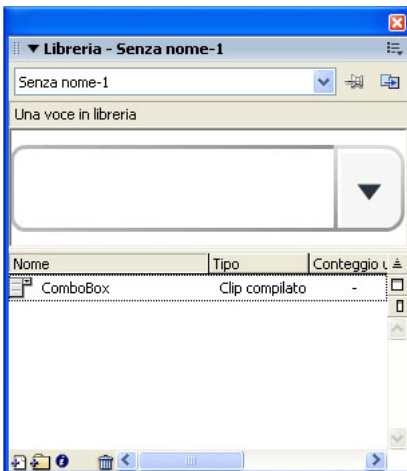
In alternativa, è possibile importare il pacchetto di classe, come illustrato nell'esempio seguente:

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

Per ulteriori informazioni, vedere [UIObject.createClassObject\(\)](#) a pagina 1408 e [Capitolo 4, “Gestione degli eventi dei componenti”](#) a pagina 69.

Componenti del pannello Libreria

Quando si aggiunge un componente a un documento, esso viene visualizzato nel pannello Libreria come simbolo di clip compilato (file SWC).



Componente ComboBox nel pannello Libreria

È possibile aggiungere più istanze di un componente trascinando nello stage l'icona del componente dalla libreria.

Per ulteriori informazioni sui clip compilati, vedere [“Informazioni su file SWC e clip compilati” a pagina 20](#).

Impostazione dei parametri dei componenti

L'aspetto e il comportamento dei componenti possono essere modificati tramite appositi parametri. Un parametro è una proprietà presente nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti. Le proprietà utilizzate più comunemente vengono visualizzate come parametri di creazione, mentre le altre devono essere impostate mediante ActionScript. Tutti i parametri impostabili durante la creazione possono essere impostati anche mediante ActionScript. I parametri impostati utilizzando ActionScript sovrascrivono qualsiasi valore impostato in fase di creazione.

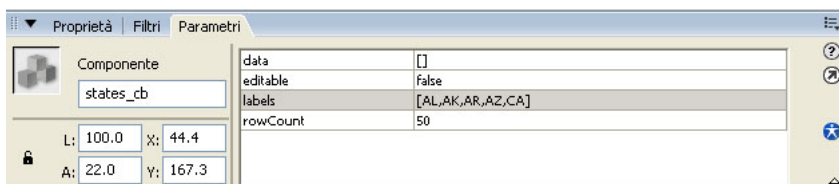
Le proprietà e i metodi utilizzati da tutti i componenti di Interfaccia utente (UI) della versione 2 sono ereditati dalle classi `UIObject` e `UIComponent`. Si tratta delle proprietà e dei metodi utilizzati da tutti i componenti, ad esempio `UIObject.setSize()`, `UIObject.setStyle()`, `UIObject.x` e `UIObject.y`. Inoltre, ciascun componente ha proprietà e metodi univoci, alcuni dei quali sono disponibili come parametri di creazione. Ad esempio, il componente `ProgressBar` dispone di una proprietà `percentComplete` (`ProgressBar.percentComplete`), mentre il componente `NumericStepper` dispone delle proprietà `nextValue` e `previousValue` (`NumericStepper.nextValue`, `NumericStepper.previousValue`).

I parametri di un'istanza di un componente possono essere impostati indifferentemente nella finestra di ispezione dei componenti o nella finestra di ispezione Proprietà.

Per immettere un nome di istanza di un componente nella finestra di ispezione Proprietà:

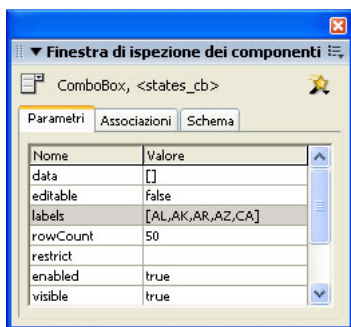
1. Selezionare Finestra > Proprietà > Proprietà.
2. Selezionare un'istanza di un componente sullo stage.
3. Immettere un nome di istanza nella casella di testo sotto la parola *Componente*.

Per semplificare la lettura del codice ActionScript, è buona norma aggiungere al nome dell'istanza un suffisso indicante il tipo di componente. In questo esempio, il nome dell'istanza è `states_cb`, dal momento che il componente è una casella combinata (combo box) contenente l'elenco degli stati degli Stati Uniti.



Per immettere i parametri di un'istanza di componente nella finestra di ispezione dei componenti:

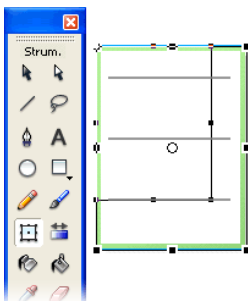
1. Selezionare Finestra > Finestra di ispezione dei componenti.
2. Selezionare un'istanza di un componente sullo stage.
3. Per immettere i parametri, fare clic sulla scheda Parametri.



4. Per immettere o visualizzare le associazioni o gli schemi per un componente, fare clic sulle schede corrispondenti. Per ulteriori informazioni, vedere “Integrazione dei dati (solo Flash Professional)” in *Uso di Flash*.

Ridimensionamento dei componenti

Per ridimensionare le istanze dei componenti, usare lo strumento Trasformazione libera oppure il metodo `setSize()`.



Ridimensionamento del componente Menu nello stage con lo strumento Trasformazione libera

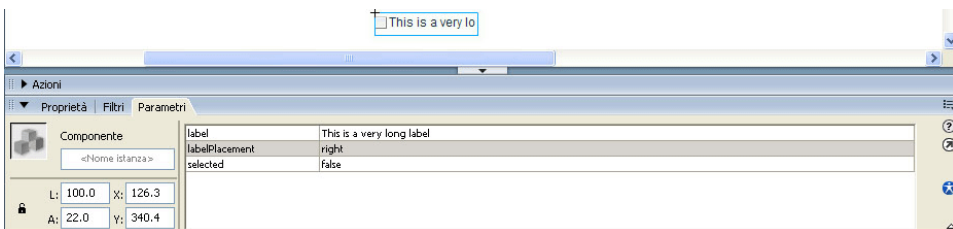
Il metodo `setSize()` può essere chiamato da qualsiasi istanza del componente da ridimensionare (vedere [UIObject.setSize\(\) a pagina 1429](#)). Il codice seguente ridimensiona il componente `TextArea` a 200 pixel di larghezza e 300 pixel di altezza:

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

NOTA

Se per regolare la larghezza e l'altezza di un componente vengono utilizzate le proprietà `ActionScript _width` e `_height`, il componente viene ridimensionato mantenendo però invariato il layout del contenuto, con il conseguente rischio di determinare una distorsione del componente durante la riproduzione del filmato.

I componenti non vengono ridimensionati automaticamente in base all'etichetta. Se l'istanza di un componente aggiunta a un documento non è sufficientemente grande da visualizzare la relativa etichetta, il testo dell'etichetta viene troncato. È necessario ridimensionare il componente in modo da rendere visibile l'intera etichetta.



Etichetta troncata del componente CheckBox

Per informazioni sul ridimensionamento dei singoli componenti, vedere le voci corrispondenti nella *Guida di riferimento dei componenti*.

Eliminazione dei componenti dai documenti Flash

Per eliminare le istanze dei componenti da un documento Flash, eliminare il componente dalla libreria rimuovendo l'icona del clip compilato. Non è sufficiente eliminare il componente dallo stage.

Per eliminare un componente da un documento:

1. Nel pannello Libreria, selezionare il simbolo del clip compilato (SWC).
2. Fare clic sul pulsante Elimina nella parte inferiore del pannello Libreria oppure selezionare Elimina dal menu Opzioni della libreria.
3. Nella finestra di dialogo Elimina, fare clic su Elimina per confermare l'eliminazione.

Uso dei suggerimenti sul codice

Quando si utilizza ActionScript 2.0, è possibile adottare la tipizzazione forte dei dati per definire una variabile basata su una classe incorporata, comprese le classi dei componenti. In questo caso, l'Editor di ActionScript visualizza i suggerimenti sul codice per la variabile. Si osservi ad esempio il seguente codice:

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

Quando si digita il punto (.) dopo `myCheckBox`, viene visualizzato un elenco di metodi e proprietà disponibili per i componenti `CheckBox`, dal momento che è stata digitata una variabile di tipo `CheckBox`. Per ulteriori informazioni, vedere “Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati” e “Uso dei suggerimenti sul codice” in *Apprendimento di ActionScript 2.0 in Flash*.

Navigazione con attivazione personalizzata del componente

Quando l'utente preme il tasto Tab per spostarsi all'interno di un'applicazione Flash o fa clic all'interno di un'applicazione, la classe `FocusManager` determina quale componente deve essere attivato. Per ulteriori informazioni, vedere [Classe FocusManager](#) nella *Guida di riferimento dei componenti*. Per attivare `FocusManager`, non è necessario aggiungerne un'istanza a un'applicazione o scrivere codice.

Se viene attivato un oggetto `RadioButton`, `FocusManager` esamina tale oggetto e tutti quelli con lo stesso valore `groupName` e imposta l'attivazione sull'oggetto con la proprietà `selected` impostata su `true`.

Ciascun componente `Window` a scelta obbligatoria contiene un'istanza di `FocusManager`; pertanto, i controlli della finestra gestita da tale componente diventano tabulazioni proprie che impediscono a un utente di passare a componenti di altre finestre premendo il tasto `Tab`.

Per creare la navigazione con attivazione in un'applicazione, impostare la proprietà `tabIndex` per tutti i componenti (inclusi i pulsanti) che devono ricevere l'attivazione. Quando l'utente preme il tasto `Tab`, la classe `FocusManager` cerca un oggetto attivato con un valore `tabIndex` maggiore del valore corrente di `tabIndex`. Dopo che la classe `FocusManager` ha raggiunto la proprietà `tabIndex` con il valore massimo, torna a 0. Nell'esempio seguente, l'oggetto `comment`, probabilmente un componente `TextArea`, viene attivato per primo, quindi viene attivato l'oggetto `okButton`:

```
var comment:mx.controls.TextArea;  
var okButton:mx.controls.Button;  
comment.tabIndex = 1;  
okButton.tabIndex = 2;
```

Per assegnare un valore di indice di tabulazione è anche possibile utilizzare il pannello `Accessibilità`.

Se sullo stage non esiste alcun elemento che abbia un valore di indice di tabulazione, `FocusManager` utilizza i livelli di profondità (*z-order*). I livelli di profondità vengono impostati principalmente in base all'ordine in cui i componenti vengono trascinati nello stage. Tuttavia, lo *z-order* finale può essere determinato anche utilizzando i comandi `Elabora > Disponi > Porta in primo piano/Porta sullo sfondo`.

Per attivare un componente in un'applicazione, chiamare la funzione `focusManager.setFocus()`.

Per creare un pulsante che venga attivato quando l'utente preme `Invio` (Windows) o `A capo` (Macintosh), impostare la proprietà `FocusManager.defaultPushButton` sull'istanza del pulsante desiderato, come nell'esempio seguente:

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

La [Classe FocusManager \(API\)](#) sostituisce il rettangolo di attivazione predefinito di Flash Player e disegna un rettangolo di attivazione personalizzato con gli angoli arrotondati.

Per ulteriori informazioni sulla creazione di uno schema di attivazione in un'applicazione Flash, vedere [Classe FocusManager](#) nella *Guida di riferimento dei componenti*.

Gestione della profondità dei componenti in un documento

Per posizionare un componente in primo piano o dietro un altro oggetto in un'applicazione, è necessario utilizzare la [Classe DepthManager](#) nella *Guida di riferimento dei componenti*. I metodi della classe DepthManager consentono di posizionare i componenti dell'interfaccia utente in un ordine *relativo* appropriato; ad esempio, una casella combinata a discesa visualizzata davanti ad altri componenti, punti di inserimento visualizzati davanti a qualsiasi altro elemento, finestre di dialogo non ancorate e così via.

DepthManager ha due funzioni principali: gestire le assegnazioni di profondità relative in qualsiasi documento e gestire le profondità riservate nella linea temporale principale per i servizi a livello di sistema, ad esempio il cursore e i suggerimenti.

Per utilizzare il componente DepthManager, chiamarne i metodi.

Il codice seguente colloca l'istanza del componente `loader` a una profondità inferiore rispetto al componente `button` (nel file SWF pubblicato esso verrà visualizzato "sotto" il pulsante, se sono sovrapposti):

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

NOTA

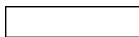
È inoltre possibile gestire le profondità relative utilizzando i Livelli e le opzioni di menu Modifica > Disponi all'interno del documento. Per i componenti valgono le stesse regole dei clip filmati, in termini di gestione della profondità di runtime utilizzando i livelli e la disposizione.

Componenti in Anteprima dal vivo

La funzione Anteprima dal vivo, attivata per impostazione predefinita, consente di esaminare sullo stage, nelle loro dimensioni approssimative, l'aspetto che i componenti assumeranno nel contenuto Flash pubblicato. L'anteprima dal vivo presenta parametri diversi a seconda dei componenti. Per informazioni sui parametri dei componenti che vengono visualizzati nell'anteprima dal vivo, vedere le voci dei singoli componenti nella *Guida di riferimento dei componenti*.

Pulsante

Un componente Button con la funzione Anteprima dal vivo attivata.



Un componente Button con la funzione Anteprima dal vivo disattivata.

I componenti visualizzati tramite la funzione Anteprima dal vivo non sono funzionali. Per verificare la funzionalità dei componenti è possibile usare il comando Controllo > Prova filmato.

Per attivare o disattivare la funzione Anteprima dal vivo:

- Selezionare Controllo > Attiva anteprima dal vivo. Un segno di spunta indica che l'opzione è attivata.

Uso di un precaricatore con i componenti

Il precaricamento riguarda il caricamento di alcuni dati per un file SWF prima che l'utente inizi a interagire con il file. Per impostazione predefinita, componenti e classi sono impostati per l'esportazione nel primo fotogramma del documento che include i componenti. Dato che componenti e classi sono i primi dati che vengono caricati, potrebbero verificarsi problemi nell'implementazione di una barra di avanzamento o nel caricamento di un'animazione. In particolare, i componenti e le classi potrebbero venire caricati prima della barra di avanzamento, mentre si desidera che la barra indichi l'avanzamento dell'operazione di caricamento di tutti i dati (incluse le classi). Le classi dovranno quindi essere caricate dopo altre parti del file SWF, ma prima che vengano utilizzati i componenti.

A questo scopo, quando si crea un precaricatore personalizzato per un'applicazione che include componenti, impostare le impostazioni di pubblicazione del file in modo che tutte le classi siano esportate nel fotogramma che include i componenti. Per visualizzare un elenco di tutti i componenti dei temi Halo e Sample le cui risorse sono impostate su Esporta nel primo fotogramma, vedere [“Modifica delle impostazioni di esportazione” a pagina 123](#).

Per modificare il fotogramma di esportazione per tutte le classi:

1. Scegliere File > Impostazioni pubblicazione.
2. Nella scheda Flash della finestra di dialogo Impostazioni di pubblicazione verificare che la versione di ActionScript sia impostata su ActionScript 2.0.
3. Fare clic sul pulsante Impostazioni a destra della versione di ActionScript.
4. In Impostazioni di ActionScript 2.0, modificare il numero della casella di testo Esporta fotogramma per le classi sul primo fotogramma nel quale il componente verrà visualizzato.

Non sarà possibile utilizzare le classi fino a quando l'indicatore di riproduzione non raggiungerà il fotogramma in cui le classi devono essere caricate. Sarà necessario caricare i componenti dopo il fotogramma in cui verranno caricate le classi perché i componenti richiedono le classi per funzionare. Se si definisce come fotogramma per le classi il fotogramma 3, non sarà possibile utilizzare alcun elemento delle classi prima che l'indicatore di riproduzione raggiunga il fotogramma 3 e i dati vengano caricati.

Per precaricare un file che utilizza i componenti, è inoltre necessario precaricare i componenti nel file SWF. A questo scopo, impostare i componenti affinché siano associati a un fotogramma diverso nel file SWF.

Per modificare il fotogramma nel quale vengono esportati i componenti:

1. Selezionare Finestra > Libreria per aprire il pannello Libreria.
2. Fare clic con il pulsante destro del mouse (Windows) o premere il tasto Ctrl (Macintosh) e fare clic su un componente della libreria.
3. Selezionare Concatenamento dal menu di scelta rapida.
4. Deselezionare Esporta nel primo fotogramma.
5. Fare clic su OK.
6. Scegliere File > Impostazioni pubblicazione.
7. Selezionare la scheda Flash e fare clic sul pulsante Impostazioni.
8. Immettere un numero nella casella di testo Esporta fotogramma per le classi e scegliere OK. Le classi verranno caricate in questo fotogramma.
9. Fare clic su OK per chiudere la finestra di dialogo Impostazioni pubblicazione.

Se i componenti non vengono caricati nel primo fotogramma, è possibile creare una barra di avanzamento personalizzata per il primo fotogramma del file SWF. Non fare riferimento a componenti del codice ActionScript, né includere componenti sullo stage fino a quando le classi per il fotogramma specificato al passaggio 7 sono state caricate.

NOTA

I componenti devono essere esportati dopo le classi ActionScript da essi utilizzate.

Informazioni sul caricamento dei componenti

Se si caricano i componenti della versione 2 in un file SWF o nel componente Loader, è possibile che i componenti non funzionino correttamente. È il caso, ad esempio, dei componenti Alert, ComboBox, DateField, Menu, MenuBar e Window.

Utilizzare la proprietà `_lockroot` per chiamare `loadMovie()` o caricare un contenuto nel componente Loader. Se si utilizza il componente Loader, aggiungere il codice seguente:

```
myLoaderComponent.content._lockroot = true;
```

Se si utilizza un clip filmato con una chiamata a `loadMovie()`, aggiungere il codice seguente:

```
myMovieClip._lockroot = true;
```

Se non si imposta `_lockroot` su `true` nel clip filmato del loader, il loader dispone dell'accesso solo alla propria libreria ma non a quella del clip filmato caricato.

La proprietà `_lockroot` è supportata da Flash Player 7. Per informazioni su questa proprietà, vedere `%{_lockroot (MovieClip._lockroot property)}%` nella *Guida di riferimento di ActionScript 2.0*.

Aggiornamento dei componenti della versione 1 all'architettura della versione 2

L'architettura dei componenti della versione 2 è conforme a diversi standard Web, ad esempio quelli relativi agli eventi (www.w3.org/TR/DOM-Level-3-Events/events.html), agli stili, alle funzioni getter/setter e così via, ed è molto diversa dall'architettura dei componenti della versione 1 forniti con Macromedia Flash MX e con i DRK rilasciati prima di Macromedia Flash MX 2004. I componenti della versione 2 vengono forniti con API diverse e sono stati scritti in ActionScript 2.0. Pertanto, l'uso combinato dei componenti della versione 1 e della versione 2 nella stessa applicazione può generare risultati imprevedibili. Per informazioni sull'aggiornamento dei componenti della versione 1 in modo che utilizzino gli standard della versione 2 per la gestione degli eventi, gli stili e l'accesso getter/setter alle proprietà al posto dei metodi, consultare il [Capitolo 6, “Creazione dei componenti”](#) a pagina 133.

Le applicazioni Flash che contengono componenti della versione 1 funzionano correttamente in Flash Player 6 e Flash Player 7, se pubblicati per Flash Player 6 o Flash Player 6 (6.0.65.0). Se si desidera aggiornare le applicazioni per utilizzarle con Flash Player 7, è necessario convertire il codice e basarlo sulla tipizzazione forte dei dati. Per ulteriori informazioni, vedere “Creazione di file di classi personalizzate” in *Apprendimento di ActionScript 2.0 in Flash*.

Gestione degli eventi dei componenti

4

Ogni componente dispone di eventi che vengono trasmessi quando l'utente interagisce con esso, ad esempio gli eventi `click` e `change`, o quando si verifica un evento significativo per il componente, ad esempio l'evento `load`. Per gestire un evento, è possibile scrivere del codice `ActionScript` che viene eseguito quando l'evento si verifica.

Ogni componente trasmette una propria serie di eventi che include gli eventi di tutte le classi dalle quali il componente eredita. Questo significa che tutti i componenti, ad eccezione dei componenti `Media`, ereditano gli eventi dalle classi `UIObject` e `UIComponent`, che sono le classi base dell'architettura della versione 2. Per visualizzare l'elenco degli eventi trasmessi da un componente, vedere le voci relative al componente e alle sue classi antenate nella *Guida di riferimento ai componenti*.

In questo capitolo viene illustrato come gestire gli eventi dei componenti mediante varie versioni di una semplice applicazione Macromedia Flash, `TipCalculator`. I file `FLA` e `SWF` vengono installati con Flash nei percorsi seguenti:

- In Windows: cartella `C:\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\TipCalculator`.
- In Macintosh: cartella `HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/TipCalculator`.

Questo capitolo contiene le seguenti sezioni:

Uso dei listener per la gestione degli eventi	70
Delega di eventi	79
Informazioni sull'oggetto evento	82
Uso del gestore di eventi <code>on()</code>	83

Uso dei listener per la gestione degli eventi

L'architettura dei componenti della versione 2 dispone di un modello di evento broadcaster/listener (il *broadcaster* a volte è anche indicato come *dispatcher*). Relativamente al modello, è importante comprendere i seguenti concetti fondamentali:

- Tutti gli eventi vengono trasmessi da un'istanza di classe di un componente (l'istanza del componente è il *broadcaster*).
- Il *listener* può essere una funzione o un oggetto. Se è un oggetto, deve disporre di una funzione di callback definita. Il listener *gestisce* l'evento, vale a dire che la funzione di callback viene eseguita quando si verifica l'evento.
- Per registrare un listener in un broadcaster, chiamare il metodo `addEventListener()` del broadcaster. Utilizzare la sintassi seguente:

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

- È possibile registrare più listener in un'istanza del componente.

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

- È possibile registrare un listener in più istanze del componente.

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

- Alla funzione del gestore viene passato un oggetto evento.

È possibile utilizzare l'oggetto evento nel corpo della funzione per recuperare le informazioni sul tipo di evento e sull'istanza che ha trasmesso l'evento. Vedere [“Informazioni sull'oggetto evento” a pagina 82](#).

- Un oggetto listener rimane attivo finché non viene rimosso esplicitamente utilizzando `EventDispatcher.removeEventListener()`. Ad esempio:

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

Uso degli oggetti listener

Per utilizzare un oggetto listener, è possibile specificare come listener l'oggetto corrente mediante la parola chiave `this`, utilizzare un oggetto esistente nell'applicazione oppure creare un nuovo oggetto.

- Nella maggior parte dei casi, utilizzare la parola chiave `this`.
Spesso è più semplice utilizzare l'oggetto corrente (`this`) come listener dal momento che l'area di validità dell'oggetto contiene i componenti che devono reagire alla trasmissione dell'evento.
- Se è opportuno, utilizzare un oggetto esistente.
Ad esempio, in un'applicazione form Flash è possibile utilizzare come oggetto listener un form contenente i componenti che reagiscono all'evento. Inserire il codice in un fotogramma della linea temporale del form.
- Se molti componenti stanno trasmettendo un evento, ad esempio l'evento `click`, e si desidera che rispondano solo determinati oggetti listener, utilizzare un nuovo oggetto listener.

Se si utilizza l'oggetto `this`, definire una funzione con lo stesso nome dell'evento che si desidera gestire. La sintassi è la seguente:

```
function eventName(evtObj:Object){  
    // Inserire qui il codice  
};
```

Se si desidera utilizzare un nuovo oggetto listener, è necessario creare l'oggetto, definire una proprietà con lo stesso nome degli eventi e assegnare la proprietà a una funzione di callback che viene eseguita quando viene trasmesso l'evento, come indicato di seguito:

```
var listenerObject:Object = new Object();  
listenerObject.eventName = function(evtObj:Object){  
    // Inserire qui il codice  
};
```

Se si desidera utilizzare un oggetto esistente, adottare la stessa sintassi di un nuovo oggetto listener, senza creare il nuovo oggetto, come illustrato di seguito:

```
existingObject.eventName = function(evtObj:Object){  
    // Inserire qui il codice  
};
```

Il parametro *evtObj* è un oggetto che viene generato automaticamente quando un evento viene attivato e passato alla funzione di callback. L'oggetto evento ha proprietà che contengono informazioni sull'evento. Per informazioni dettagliate, vedere [“Informazioni sull'oggetto evento” a pagina 82](#).

Infine, chiamare il metodo `addEventListener()` dall'istanza di componente che trasmette l'evento. Il metodo `addEventListener()` accetta due parametri: una stringa che indica il nome dell'evento e un riferimento all'oggetto listener.

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

Di seguito è riportato l'intero segmento di codice che è possibile copiare e incollare. Assicurarsi di sostituire tutto il codice in corsivo con i valori effettivi; è possibile utilizzare *listenerObject* e *evtObj* o altri identificatori validi, ma è obbligatorio sostituire *nomeEvento* con il nome dell'evento.

```
var listenerObject:Object = new Object();
listenerObject.eventName = function(evtObj:Object){
    // Il codice che si trova in questa posizione viene eseguito
    // quando viene attivato l'evento
};
componentInstance.addEventListener("eventName", listenerObject);
```

Nel seguente segmento di codice la parola chiave `this` viene utilizzata come oggetto listener:

```
function eventName(evtObj:Object){
    // Il codice che si trova in questa posizione viene eseguito
    // quando viene attivato l'evento
}
componentInstance.addEventListener("eventName", this);
```

Il metodo `addEventListener()` può essere chiamato da qualsiasi istanza di componente in quanto è incorporato in tutti i componenti della classe `EventDispatcher`. Un "mix-in" è una classe che fornisce funzioni specifiche che potenziano il comportamento di un'altra classe. Per ulteriori informazioni, vedere “`EventDispatcher.addEventListener()`” nella *Guida di riferimento dei componenti*.

Per informazioni sugli eventi inviati da un particolare componente, vedere la relativa voce nella *Guida di riferimento dei componenti*. Ad esempio, un elenco degli eventi del componente `Button` è disponibile nella sezione relativa al componente `Button` oppure selezionando ? > Guida di riferimento dei componenti > Componente `Button` > Classe `Button` > Riepilogo degli eventi validi per la classe `Button`.

Per registrare un oggetto listener in un file Flash (FLA):

1. In Flash, selezionare File > Nuovo e creare un documento Flash nuovo.
2. Trascinare un componente Button nello stage dal pannello Componenti.
3. Nella finestra di ispezione Proprietà, immettere il nome di istanza **myButton**.
4. Trascinare il componente TextInput nello stage dal pannello Componenti.
5. Nella finestra di ispezione Proprietà, immettere il nome di istanza **myText**.
6. Selezionare il fotogramma 1 nella linea temporale.
7. Selezionare Finestra > Azioni.
8. Nel pannello Azioni, inserire il codice seguente:

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

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

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

La proprietà `target` dell'oggetto evento `evt` è un riferimento all'istanza che trasmette l'evento. Questo codice visualizza il valore della proprietà `target` nel componente `TextInput`.

Per registrare un oggetto listener in un file di classe (AS):

1. Aprire il file `TipCalculator.fla` dal percorso specificato in [“Operazioni con i componenti” a pagina 53](#).
2. Aprire il file `TipCalculator.as` dal percorso specificato in [“Operazioni con i componenti” a pagina 53](#).
3. Nel file FLA, selezionare `form1` e visualizzare il nome di classe `TipCalculator` nella finestra di ispezione Proprietà.

Si tratta del collegamento tra il form e il file di classe. Tutto il codice per l'applicazione è incluso nel file `TipCalculator.as`. Il form assume le proprietà e i comportamenti definiti dalla classe a esso assegnata.

4. Nel file AS, scorrere fino alla riga 25, `public function onLoad():Void`.

La funzione `onLoad()` viene eseguita quando il form viene caricato in Flash Player. Nel corpo della funzione, l'istanza `TextInput subtotal` e le tre istanze `RadioButton`, `percentRadio15`, `percentRadio18` e `percentRadio20`, chiamano il metodo `addEventListener()` per registrare un listener in un evento.

5. Esaminare la riga 27, `subtotal.addEventListener("change", this)`.

Quando si chiama `addEventListener()`, passare a questa riga due parametri. Il primo è una stringa che indica il nome dell'evento che viene trasmesso, in questo caso "change". Il secondo è un riferimento a un oggetto o a una funzione che gestisce l'evento. In questo caso, il parametro è la parola chiave `this`, che fa riferimento a un'istanza del file di classe, ovvero a un oggetto. Flash cerca quindi nell'oggetto una funzione con il nome dell'evento.

6. Esaminare la riga 63, `public function change(event:Object):Void`.

Si tratta della funzione che viene eseguita quando viene modificata l'istanza `TextInput subtotal`.

7. Selezionare il file `TipCalculator.fla`, quindi Controllo > Prova filmato per provare il file.

Uso della funzione di callback `handleEvent`

È anche possibile utilizzare gli oggetti listener che supportano una funzione `handleEvent`. Indipendentemente dal nome dell'evento trasmesso, viene chiamato il metodo `handleEvent` dell'oggetto listener. Per gestire più eventi, è necessario utilizzare un'istruzione `if..else` o `switch`. Ad esempio, il codice seguente utilizza un'istruzione `if..else` per gestire gli eventi `click` e `change`:

```
// Definisce la funzione handleEvent
// Passa evt come parametro dell'oggetto evento

function handleEvent(evt){
    // Verifica se l'evento era click
    if (evt.type == "click"){
        // Esegue un'operazione se l'evento era click
    } else if (evt.type == "change"){
        // Esegue un'operazione diversa se l'evento era change
    }
};

// Registra l'oggetto listener in
// due diverse istanze di componente.
// Poiché la funzione è definita
// sull'oggetto "this", il listener è this.

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

Uso delle funzioni listener

A differenza della sintassi di `handleEvent`, i diversi eventi possono essere gestiti da funzioni listener diverse. Quindi, anziché inserire le istruzioni `if e else if in myHandler`, è sufficiente definire `myChangeListener` per l'evento `change` e `myScrollHandler` per l'evento `scroll` e registrarli come indicato di seguito:

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

Per utilizzare una funzione listener, occorre dapprima definire una funzione:

```
function myFunction:Function(evtObj:Object){
    // Inserire qui il codice
}
```

SUGGERIMENTO

Il parametro `evtObj` è un oggetto che viene generato automaticamente quando un evento viene attivato e passato alla funzione. L'oggetto evento ha proprietà che contengono informazioni sull'evento. Per informazioni dettagliate, vedere ["Informazioni sull'oggetto evento"](#) a pagina 82.

Chiamare quindi il metodo `addEventListener()` dall'istanza di componente che trasmette l'evento. Il metodo `addEventListener()` accetta due parametri: una stringa che indica il nome dell'evento e un riferimento alla funzione.

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

Il metodo `addEventListener()` può essere chiamato da qualsiasi istanza di componente in quanto è incorporato in tutti i componenti UI della classe `EventDispatcher`. Per ulteriori informazioni, vedere [EventDispatcher.addEventListener\(\)](#).

Per informazioni sugli eventi inviati da un componente, vedere la relativa voce ai singoli componenti nella *Guida di riferimento dei componenti*.

Per registrare un oggetto listener in un file Flash (FLA):

1. In Flash, selezionare File > Nuovo e creare un documento Flash nuovo.
2. Trascinare un componente List nello stage dal pannello Componenti.
3. Nella finestra di ispezione Proprietà, immettere il nome di istanza **myList**.
4. Selezionare il fotogramma 1 nella linea temporale.
5. Selezionare Finestra > Azioni.

6. Nel pannello Azioni, inserire il codice seguente:

```
// Dichiarare le variabili
var myList:mx.controls.List;
var myHandler:Function;

// Aggiunge voci all'elenco
myList.addItem("Bird");
myList.addItem("Dog");
myList.addItem("Fish");
myList.addItem("Cat");
myList.addItem("Ape");
myList.addItem("Monkey");

// Definisce la funzione myHandler
function myHandler(eventObj:Object){

    // Utilizza il parametro eventObj
    // per acquisire il tipo di evento
    if (eventObj.type == "change"){
        trace("The list changed");
    } else if (eventObj.type == "scroll"){
        trace("The list was scrolled");
    }
}

// Registra la funzione myHandler con myList.
// Quando viene selezionata una voce (viene attivato l'evento change) o
// si scorre l'elenco, viene eseguita la funzione myHandler.
myList.addEventListener("change", myHandler);
myList.addEventListener("scroll", myHandler);
```

NOTA

La proprietà type dell'oggetto evento, evt, è un riferimento al nome dell'evento.

7. Selezionare Controllo > Prova filmato; quindi selezionare un elemento dall'elenco e scorrere l'elenco per visualizzare i risultati nel pannello Output.

ATTENZIONE

In una funzione listener, la parola chiave this fa riferimento all'istanza del componente che chiama addEventListener(), non alla linea temporale o alla classe in cui viene definita la funzione. Tuttavia, è possibile delegare la funzione listener a un'area di validità diversa mediante la classe Delegate. Vedere [“Delega di eventi” a pagina 79](#). Per un esempio di assegnazione di un'area di validità a una funzione, consultare la sezione successiva.

Informazioni sull'area di validità dei listener

L'*area di validità* è l'oggetto all'interno del quale viene eseguita una funzione. I riferimenti alle variabili all'interno della funzione vengono considerati come proprietà dell'oggetto. Per specificare l'area di validità di un listener è possibile utilizzare la classe Delegate. Per ulteriori informazioni, vedere [“Delega di eventi” a pagina 79](#).

Come illustrato in precedenza, è possibile registrare un listener con un'istanza di componente chiamando `addEventListener()`. Tale metodo acquisisce due parametri: una stringa che indica il nome dell'evento e un riferimento a un oggetto o a una funzione. Nella seguente tabella viene indicata l'area di validità di ogni tipo di parametro:

Tipo di listener	Area di validità
Oggetto	Oggetto listener.
Funzione	Istanza di componente che trasmette l'evento.

Se al metodo `addEventListener()` viene passato un oggetto, la funzione di callback assegnata a tale oggetto o la funzione in esso definita viene chiamata nell'area di validità dell'oggetto. Questo significa che la parola chiave `this`, quando viene utilizzata nella funzione di callback, fa riferimento all'oggetto listener, come indicato di seguito:

```
var lo:Object = new Object();
lo.click = function(evt){
    // this fa riferimento all'oggetto lo
    trace(this);
}
myButton.addEventListener("click", lo);
```

Tuttavia, se al metodo `addEventListener()` viene passata una funzione, essa viene chiamata nell'area di validità dell'istanza di componente che chiama `addEventListener()`. Pertanto la parola chiave `this`, quando viene utilizzata nella funzione, fa riferimento all'istanza di componente broadcaster. Questo può costituire un problema se si definisce la funzione in un file di classe, in quanto non è possibile accedere alle proprietà e ai metodi del file di classe con i percorsi previsti perché `this` non punta a un'istanza della classe. Per risolvere questo problema, delegare una funzione all'area di validità corretta mediante la classe Delegate. Vedere [“Delega di eventi” a pagina 79](#).

Nel codice seguente viene illustrata l'assegnazione di un'area di validità a una funzione quando questa viene passata a `addEventListener()` in un file di classe. Per utilizzare questo codice, copiarlo in un file `ActionScript (AS)` denominato `Cart.as`. Creare un file `Flash (FLA)` con un componente `Button`, `myButton`, e un componente `DataGrid`, `myGrid`. Selezionare entrambi i componenti sullo stage e premere `F8` per convertirli in un nuovo simbolo denominato `Cart`. Nelle proprietà di concatenamento del simbolo `Cart`, assegnare al simbolo la classe `Cart`.

```
class Cart extends MovieClip {

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

    function myHandler(eventObj:Object){

        // Utilizza il parametro eventObj
        // per acquisire il tipo di evento.
        if (eventObj.type == "click"){

            /* Invia il valore di this al pannello Output.
            Poiché myHandler è una funzione non definita
            in un oggetto listener, this è un riferimento
            all'istanza di componente nella quale viene registrata myHandler
            (myButton). Inoltre, poiché this non fa riferimento a
            un'istanza della classe Cart, myGrid è undefined.
            */
            trace("this: " + this);
            trace("myGrid: " + myGrid);
        }
    }

    // Registra la funzione myHandler con myButton
    // Quando si fa clic sul pulsante, myHandler viene eseguita

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

Delega di eventi

Per delegare gli eventi a specifiche aree di validità e funzioni, è possibile importare la classe `Delegate` negli script o nelle classi. Vedere “Classe Delegate” nella *Guida di riferimento dei componenti*. Per importare la classe `Delegate`, utilizzare la sintassi seguente:

```
import mx.utils.Delegate;
compInstance.addEventListener("eventName", Delegate.create(scopeObject,
    function));
```

Il parametro `scopeObject` specifica l'area di validità nella quale viene chiamato il parametro `function` specificato.

La chiamata di `Delegate.create()` viene utilizzata più comunemente in due casi:

- Per inviare lo stesso evento a due funzioni diverse.
Consultare la sezione successiva.
- Per chiamare funzioni all'interno dell'area di validità della classe in cui sono contenute.
Quando si passa una funzione come parametro a `addEventListener()`, la funzione viene chiamata nell'area di validità dell'istanza di componente broadcaster, non dell'oggetto nel quale viene dichiarata. Vedere [“Delega dell'area di validità di una funzione” a pagina 81](#).

Delega di eventi alle funzioni

La chiamata di `Delegate.create()` risulta utile in presenza di due componenti che trasmettono eventi con lo stesso nome. Se, ad esempio, si dispone di una casella di controllo e di un pulsante, occorre utilizzare l'istruzione `switch` per le informazioni ricevute dalla proprietà `eventObject.target` in modo da individuare il componente dal quale viene trasmesso l'evento `click`.

Per utilizzare il codice seguente, posizionare sullo stage una casella di controllo denominata `myCheckBox_chb` e un pulsante denominato `myButton_btn`. Selezionare entrambe le istanze e premere F8 per creare un nuovo simbolo. Se la finestra di dialogo è in modalità di base, fare clic su Avanzate e selezionare Esporta per ActionScript. Immettere `Cart` nella casella di testo Classe AS 2.0. Nella finestra di ispezione Proprietà, impostare il nome di istanza desiderato per il nuovo simbolo. Il simbolo viene associato alla classe `Cart` e un'istanza del simbolo diventa un'istanza di questa classe.

```
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:
            // Invia il nome di istanza del broadcaster
            // e il tipo di evento al pannello Output
            trace(eventObj.target + ": " + eventObj.type);
            break;
        case myCheckBox_chb:
            trace(eventObj.target + ": " + eventObj.type);
            break;
    }
}
}

```

Di seguito è riportato lo stesso file di classe (Cart.as) modificato per l'uso di 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));
    }

    // Due funzioni separate gestiscono gli eventi

    function chb_onClick(eventObj:Object) {
        // Invia il nome di istanza del broadcaster
        // e il tipo di evento al pannello Output
        trace(eventObj.target + ": " + eventObj.type);
        // Invia il percorso assoluto del simbolo
        // associato alla classe Cart
        // nel file FLA al pannello Output
        trace(this)
    }

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

```


Delega dell'area di validità di una funzione

Il metodo `addEventListener()` richiede due parametri: il nome di un evento e un riferimento a un listener. Il listener può essere un oggetto o una funzione. Se si passa un oggetto, la funzione di callback assegnata all'oggetto viene chiamata nell'area di validità dell'oggetto. Tuttavia, se si passa una funzione, tale funzione viene chiamata nell'area di validità dell'istanza di componente che chiama `addEventListener()`. Per ulteriori informazioni, vedere [“Informazioni sull'area di validità dei listener” a pagina 77](#).

Poiché la funzione viene chiamata nell'area di validità dell'istanza del broadcaster, la parola chiave `this` nel corpo della funzione punta all'istanza del broadcaster e non alla classe che contiene la funzione. Di conseguenza, non è possibile accedere alle proprietà e ai metodi della classe contenente la funzione. Per accedere alle proprietà e ai metodi di tale classe, utilizzare la classe `Delegate` per delegare l'area di validità della funzione alla classe che la contiene.

Nell'esempio seguente viene utilizzato lo stesso approccio della sezione precedente, con una variante nel file di classe `Cart.as`:

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

class Cart {

    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    // Definisce una variabile accessibile
    // dalla funzione chb_onClick
    var i:Number = 10

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

    function chb_onClick(eventObj:Object) {
        // Si prevede di poter accedere
        // alla variabile i e all'output 10.
        // Tuttavia, this invia undefined
        // al pannello Output perché
        // la funzione non è limitata
        // all'istanza Cart nella quale è definita la variabile i.
        trace(i);
    }
}
```

Per accedere alle proprietà e ai metodi della classe `Cart`, chiamare `Delegate.create()` come secondo parametro di `addEventListener()`, come indicato di seguito:

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

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;
    // Definisce una variabile accessibile
    // dalla funzione chb_onClick
    var i:Number = 10

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

    function chb_onClick(eventObj:Object) {
        // Invia 10 al pannello Output
        // perché la funzione è limitata
        // all'istanza Cart
        trace(i);
    }
}
```

Informazioni sull'oggetto evento

L'oggetto evento è un'istanza della classe `Object` di `ActionScript` e dispone delle seguenti proprietà che contengono le informazioni sull'evento.

Proprietà	Descrizione
<code>type</code>	Una stringa che indica il nome dell'evento.
<code>target</code>	Un riferimento all'istanza di componente che trasmette l'evento.

Le eventuali proprietà aggiuntive di un evento sono elencate alla voce dell'evento nel Dizionario dei componenti.

L'oggetto evento viene generato automaticamente quando un evento viene attivato e passato alla funzione di callback dell'oggetto listener o alla funzione del listener.

È possibile utilizzare l'oggetto evento all'interno della funzione per accedere al nome dell'evento trasmesso o al nome di istanza del componente che ha trasmesso l'evento. Dal nome di istanza è possibile accedere alle altre proprietà del componente. Ad esempio, nel codice seguente viene utilizzata la proprietà `target` dell'oggetto evento `evtObj` per accedere alla proprietà `label` dell'istanza `myButton` e inviare il valore al pannello Output:

```
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);
```

Uso del gestore di eventi `on()`

È possibile assegnare il gestore di eventi `on()` all'istanza di un componente con metodi analoghi a quelli per l'assegnazione di un gestore a un pulsante oppure a un clip filmato. Un gestore di eventi `on()` può essere utile per eseguire una semplice verifica, ma per tutte le applicazioni è consigliabile utilizzare i listener di eventi. Per ulteriori informazioni, vedere [“Uso dei listener per la gestione degli eventi” a pagina 70](#).

Quando si utilizza la parola chiave `this` all'interno di un gestore `on()` associato direttamente a un componente (assegnato all'*istanza* del componente nel pannello Azioni), `this` fa riferimento all'istanza del componente. Ad esempio, il codice seguente, associato all'istanza del componente Button `myButton`, visualizza `"_level0.myButton"` nel pannello Output:

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

Per utilizzare il gestore di eventi `on()`:

1. Trascinare nello stage un componente di interfaccia utente.
Ad esempio, trascinare nello stage un componente Button.
2. Selezionare il componente sullo stage e aprire il pannello Azioni.

3. Aggiungere il gestore `on()` al pannello Azioni pannello nel formato seguente:

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

Ad esempio:

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

Flash esegue il codice all'interno del gestore `on()` quando si verifica l'evento per il gestore `on()` (in questo caso, quando viene fatto clic su un pulsante).

4. Selezionare Controllo > Prova filmato, quindi fare clic sul pulsante per visualizzare l'output.

Personalizzazione dei componenti

Quando si utilizzano i componenti, può essere utile cambiarne l'aspetto a seconda del tipo di applicazione. Per personalizzare l'aspetto dei componenti, sia individualmente che insieme, sono disponibili i tre approcci seguenti:

Stili I componenti dell'interfaccia utente (UI) dispongono di proprietà di stile che consentono di impostare la visualizzazione di alcuni aspetti. Per ogni componente è disponibile un insieme specifico di proprietà di stile modificabili e non tutti gli aspetti visivi possono essere modificati mediante l'impostazione di uno stile. Per ulteriori informazioni, vedere [“Uso degli stili per personalizzare il testo e il colore dei componenti” a pagina 86](#).

Skin Uno *skin* include l'insieme dei simboli che compongono la visualizzazione grafica di un componente. L'*associazione di skin* rappresenta il processo di modifica dell'aspetto di un componente attraverso il cambiamento o la sostituzione della sua immagine di origine. Uno skin può essere costituito da un piccolo tassello, come il margine o l'angolo di un bordo, oppure da un elemento composito, ad esempio l'intera immagine di un pulsante nello stato Su (ossia lo stato in cui si trova quando non è premuto). Uno skin può anche essere un simbolo privo di grafica, contenente codice che disegna una parte del componente. Alcuni aspetti di un componente, che non è possibile impostare utilizzando le proprietà di stile, possono essere impostati mediante la modifica dello skin. Per ulteriori informazioni, vedere [“Informazioni sull'associazione di skin ai componenti” a pagina 101](#).

Temi Un tema è composto da un insieme di stili e skin che è possibile salvare come file FLA e applicare a un altro documento. Per ulteriori informazioni, vedere [“Informazioni sui temi” a pagina 115](#).

Questo capitolo contiene le seguenti sezioni:

Uso degli stili per personalizzare il testo e il colore dei componenti	86
Informazioni sull'associazione di skin ai componenti	101
Informazioni sui temi	115
Combinazione di skin e di stili per personalizzare un componente	125

Uso degli stili per personalizzare il testo e il colore dei componenti

Le proprietà di stile disponibili in Flash possono essere modificate per ogni componente dell'interfaccia utente. Nella documentazione di ogni specifico componente, è presente una tabella con l'elenco degli stili modificabili per quel componente; ad esempio, in “Uso degli stili con il componente Accordion” nella *Guida di riferimento dei componenti* è possibile visualizzare una tabella degli stili del componente Accordion. I componenti dell'interfaccia utente ereditano inoltre i metodi `setStyle()` e `getStyle()` dalla classe `UIObject`. Vedere `UIObject.setStyle()` e `UIObject.getStyle()`. Per impostare e ottenere i valori delle proprietà di stile per un'istanza di componente, è possibile utilizzare i metodi `setStyle()` e `getStyle()`, descritti più avanti in [“Impostazione degli stili in un'istanza di componente” a pagina 89](#).

NOTA

Non è possibile impostare gli stili per i componenti Media.

Uso di dichiarazioni di stile e temi

In senso più generale, gli stili sono organizzati in *dichiarazioni* di stile nelle quali è possibile controllare i valori delle proprietà di stile nelle diverse istanze di un componente. Una dichiarazione di stile è un oggetto creato dalla classe `CSSStyleDeclaration` e le relative proprietà sono le impostazioni di stile che è possibile assegnare ai componenti. In ActionScript le dichiarazioni di stile seguono il modello applicato da CSS (Cascading Style Sheet) alle pagine HTML. Per le pagine HTML, è possibile creare un file foglio di stile per definire le proprietà di stile per il contenuto di un gruppo di pagine HTML. Per i componenti, è possibile controllare l'aspetto dei componenti in un documento Flash creando un oggetto dichiarazione di stile e aggiungendo le proprietà di stile a tale oggetto.

Le dichiarazioni di stile sono inoltre organizzate in *temi*. In Flash sono disponibili due temi visivi per i componenti: Halo (HaloTheme.fla) e Sample (SampleTheme.fla). Il *tema* è un insieme di stili ed elementi grafici che controlla l'aspetto dei componenti di un documento. Ogni tema fornisce stili ai componenti. Gli stili utilizzati da ogni componente dipendono in parte dal tema utilizzato dal documento. Alcuni stili, come `defaultIcon`, vengono utilizzati dai componenti associati indipendentemente dal tema applicato al documento. Altri stili, come `themeColor` e `symbolBackgroundColor`, vengono utilizzati dai componenti solo se è in uso il tema corrispondente. Ad esempio, `themeColor` viene utilizzato solo se è in uso il tema Halo, mentre `symbolBackgroundColor` viene utilizzato solo se è in uso il tema Sample. Per determinare le proprietà di stile che si possono impostare per un componente, individuare il tema assegnato al componente. La *Guida di riferimento dei componenti* indica se le singole proprietà di stile possono essere applicate a uno o entrambi i temi forniti. Per ulteriori informazioni, vedere [“Informazioni sui temi” a pagina 115](#).

Nozioni fondamentali sulle impostazioni di stile

Quando si utilizzano stili e dichiarazioni di stile, è possibile notare che esistono varie modalità di impostazione degli stili, ad esempio a livello di tema, classe, dichiarazione di stile, proprietà di stile oppure a livello globale. Alcune proprietà di stile possono inoltre essere ereditate da un componente principale; ad esempio, un pannello Accordion secondario può ereditare la modalità di gestione di un carattere dal componente Accordion. Di seguito vengono descritti alcuni punti chiave sul comportamento degli stili:

Dipendenza dal tema Le proprietà di stile che si possono impostare su un componente particolare sono determinate dal tema corrente. Per impostazione predefinita, per i componenti di Flash è previsto l'uso del tema Halo, tuttavia è disponibile anche un tema Sample. Quando si visualizza una tabella delle proprietà di stile, come quella del componente Button in “Uso degli stili con il componente Button” nella *Guida di riferimento dei componenti*, osservare quale tema supporta lo stile desiderato: "Halo", "Sample" o "Entrambi" (indica che entrambi i temi supportano quella proprietà di stile). Per cambiare il tema corrente, vedere [“Passaggio tra temi” a pagina 115](#).

Ereditarietà Non è possibile impostare l'ereditarietà in ActionScript. L'elemento secondario di un componente viene progettato in modo da ereditare o meno uno stile dal componente principale.

Fogli di stile globali A differenza del comportamento di CSS per i documenti HTML, in Flash non è supportato il meccanismo “a cascata” per i documenti Flash. Tutti gli oggetti dichiarazione foglio di stile vengono definiti a livello di applicazione (globale).

Precedenza Se lo stile di un componente viene impostato in più modi, ad esempio se `textColor` è impostato sia a livello globale sia a livello di istanza del componente, viene utilizzato automaticamente il primo stile rilevato da Flash in base all'ordine elencato in [“Uso degli stili globale, personalizzato e classe nello stesso documento” a pagina 97](#).

Impostazione di stili

L'esistenza delle proprietà di stile, la loro organizzazione all'interno delle dichiarazioni di stile e la più ampia organizzazione delle dichiarazioni di stile e della grafica in temi consentono di personalizzare un componente nei modi seguenti:

- Impostando gli stili in un'istanza di un componente.

È possibile modificare le proprietà di colore e testo di un'unica istanza di componente. Questo metodo può risultare efficace in alcune situazioni, tuttavia può richiedere un certo dispendio di tempo nei casi in cui sia necessario impostare le singole proprietà di tutti i componenti presenti in un documento.

Per ulteriori informazioni, vedere [“Impostazione degli stili in un'istanza di componente” a pagina 89](#).
- Modificando la dichiarazione di stile globale che imposta gli stili di tutti i componenti di un documento.

Se si desidera applicare un aspetto omogeneo a un intero documento, è possibile creare gli stili nella dichiarazione di stile globale.

Per ulteriori informazioni, vedere [“Impostazione di stili globali” a pagina 91](#).
- Creando dichiarazioni di stile personalizzate e applicandole a varie istanze di componente.

Se si desidera che più gruppi di componenti di un documento condividano lo stesso stile, è possibile creare dichiarazioni di stile personalizzate da applicare ai componenti desiderati.

Per ulteriori informazioni, vedere [“Impostazione di stili personalizzati per gruppi di componenti” a pagina 92](#).
- Creando dichiarazioni di stile classe predefinite.

È possibile definire una dichiarazione di stile classe predefinita, in modo che tutte le istanze di una classe abbiano lo stesso aspetto predefinito.

Per ulteriori informazioni, vedere [“Impostazione di stili per una classe di componente” a pagina 94](#).

- Utilizzando gli stili ereditari per impostare gli stili dei componenti di una parte di documento.

I valori delle proprietà di stile impostati per i contenitori vengono ereditati dai componenti in essi contenuti.

Per ulteriori informazioni, vedere [“Impostazione di stili ereditari in un contenitore” a pagina 95](#).

Le modifiche apportate alle proprietà di stile non vengono visualizzate in Flash quando si visualizzano i componenti nello stage con la funzione Anteprima dal vivo. Per ulteriori informazioni, vedere [“Componenti in Anteprima dal vivo” a pagina 64](#).

Impostazione degli stili in un'istanza di componente

È possibile creare istruzioni di ActionScript per impostare e ottenere proprietà di stile in qualsiasi istanza di componente. I metodi `UIObject.setStyle()` e `UIObject.getStyle()` possono essere chiamati direttamente da qualunque componente dell'interfaccia utente. La sintassi seguente specifica una proprietà e il valore di un'istanza di componente:

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

Ad esempio, il codice seguente imposta i colori di risalto di un'istanza Button denominata `myButton` che utilizza il tema Halo:

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

NOTA

Se il valore è una stringa, è necessario racchiuderlo tra virgolette.

È inoltre possibile accedere agli stili direttamente come proprietà (ad esempio, `myButton.color = 0xFF00FF`).

Le proprietà di stile impostate in un'istanza di componente mediante il metodo `setStyle()` hanno la priorità più elevata e sostituiscono tutte le altre impostazioni di stile basate su dichiarazione di stile o tema. L'impostazione di un numero elevato di proprietà tramite il metodo `setStyle()` per un'unica istanza di componente causa tuttavia il rallentamento del rendering del componente in fase di runtime. Se si utilizza `UIObject.createClassObject()` nella *Guida di riferimento dei componenti* e si inseriscono le impostazioni di stile nel parametro *initObject*, è possibile velocizzare il rendering di un componente personalizzato con codice ActionScript, che definisce le proprietà di stile durante la creazione dell'istanza del componente. Ad esempio, con un componente `ComboBox` nella libreria del documento corrente, il codice seguente crea un'istanza di casella combinata con nome `my_cb` e imposta il testo in corsivo e allineato a destra all'interno della casella combinata:

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

NOTA

Se si desidera modificare più proprietà o le proprietà di più istanze di un componente, è possibile creare uno stile personalizzato. Il rendering dell'istanza di un componente che utilizza uno stile personalizzato per più proprietà è più veloce rispetto all'istanza di un componente con più chiamate `setStyle()`. Per ulteriori informazioni, vedere ["Impostazione di stili personalizzati per gruppi di componenti" a pagina 92](#).

Per impostare o modificare una proprietà per una singola istanza di componente che utilizza il tema Halo:

1. Selezionare l'istanza del componente sullo stage.
2. Nella finestra di ispezione Proprietà, immettere **myComponent** come nome di istanza.
3. Aprire il pannello Azioni e selezionare Scena 1, quindi selezionare Livello 1: Fotogramma 1.
4. Immettere il codice seguente per cambiare il colore dell'istanza e impostarlo sull'arancione:
`myComponent.setStyle("themeColor", "haloOrange");`
5. Selezionare Controllo > Prova filmato per visualizzare le modifiche.

Per un elenco degli stili supportati da un componente specifico, vedere la voce relativa al componente nella *Guida di riferimento dei componenti*.

Per creare l'istanza di un componente e impostare più proprietà contemporaneamente utilizzando `ActionScript`:

1. Trascinare un componente nella libreria.
2. Aprire il pannello Azioni e selezionare Scena 1, quindi selezionare Livello 1: Fotogramma 1.
3. Per creare un'istanza del componente e impostare le relative proprietà, immettere la sintassi seguente:

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

Di conseguenza, ad esempio per un componente `Button` nella libreria, il codice `ActionScript` seguente crea un'istanza di `my_button` alla profondità 1 con gli stili di testo impostati su viola e corsivo:

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

Per ulteriori informazioni, vedere `UIObject.createClassObject()`.

4. Selezionare **Controllo > Prova filmato** per vedere le modifiche.

Per un elenco degli stili supportati da un componente specifico, vedere la voce relativa al componente nella *Guida di riferimento dei componenti*.

Impostazione di stili globali

Per impostazione predefinita, tutti i componenti seguono la dichiarazione di stile globale, finché a un componente non viene assegnata un'altra dichiarazione di stile, come descritto in [“Impostazione di stili personalizzati per gruppi di componenti” a pagina 92](#). La dichiarazione di stile globale è assegnata a tutti i componenti Flash forniti con la versione 2 di Macromedia Component Architecture. L'oggetto `_global` dispone di una proprietà `style` (`_global.style`) che rappresenta un'istanza di `CSSStyleDeclaration` e funziona come dichiarazione di stile globale. Se si modifica il valore di una proprietà di stile nella dichiarazione di stile globale, la modifica viene applicata a tutti i componenti del documento Flash.

ATTENZIONE

Alcuni stili vengono impostati nell'istanza `CSSStyleDeclaration` della classe di un componente, ad esempio lo stile `backgroundColor` dei componenti `TextArea` e `TextInput`. Dal momento che, quando vengono determinati i valori di stile, la dichiarazione di stile classe ha la priorità rispetto alla dichiarazione di stile globale, l'impostazione di `backgroundColor` nella dichiarazione di stile globale non ha alcun effetto sui componenti `TextArea` e `TextInput`. Per ulteriori informazioni sulla priorità degli stili, vedere [“Uso degli stili globale, personalizzato e classe nello stesso documento” a pagina 97](#). Per ulteriori informazioni sulla modifica dell'istanza `CSSStyleDeclaration` della classe di un componente, vedere [“Impostazione di stili per una classe di componente” a pagina 94](#).

Per modificare una o più proprietà nella dichiarazione di stile `_global`:

1. Verificare che il documento contenga almeno un'istanza del componente.
Per ulteriori informazioni, vedere [“Aggiunta di componenti ai documenti Flash” a pagina 54](#).
2. Selezionare nella linea temporale un fotogramma nel quale, o prima del quale, sono visualizzati i componenti.
3. Inserire nel pannello Azioni il codice seguente per modificare le proprietà nella dichiarazione di stile globale. Elencare solo le proprietà di cui si desidera modificare i valori, come indicato di seguito:

```
_global.style.setStyle("color", 0xCC6699);  
_global.style.setStyle("themeColor", "haloBlue")  
_global.style.setStyle("fontSize",16);  
_global.style.setStyle("fontFamily" , "_serif");
```
4. Selezionare Controllo > Prova filmato per visualizzare le modifiche.

Impostazione di stili personalizzati per gruppi di componenti

Per specificare un insieme specifico di proprietà per gruppi di componenti di un documento Flash, è possibile creare dichiarazioni di stile personalizzate. Oltre alla proprietà `style` dell'oggetto `_global`, descritta in [“Impostazione di stili globali” a pagina 91](#), che determina la dichiarazione di stile predefinita per un intero documento Flash, l'oggetto `_global` dispone inoltre di una proprietà `styles` che corrisponde a un elenco delle dichiarazioni di stile personalizzate disponibili. È quindi possibile creare una dichiarazione di stile come una nuova istanza dell'oggetto `CSSStyleDeclaration`, assegnarvi un nome di stile personalizzato e inserirla nell'elenco `_global.styles`. Successivamente, si specificano le proprietà e i valori per lo stile e si assegna il nome di stile alle istanze del componente che devono condividere lo stesso aspetto.

Si tenga presente che quando si assegna il nome di stile a un'istanza di un componente, il componente risponde solo alle proprietà di stile supportate dal componente stesso. Per un elenco delle proprietà di stile supportate da ogni componente, vedere le voci relative ai singoli componenti nella *Guida di riferimento dei componenti*.

Per modificare un formato di stile personalizzato, utilizzare la sintassi seguente:

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

Le impostazioni di stile personalizzato hanno la priorità sulle impostazioni di stile classe, ereditato e globale. Per un elenco relativo alla priorità degli stili, vedere [“Uso degli stili globale, personalizzato e classe nello stesso documento” a pagina 97](#).

Per creare una dichiarazione di stile personalizzata per un gruppo di componenti:

1. Aggiungere almeno un componente nello stage.

Per ulteriori informazioni, vedere [“Aggiunta di componenti ai documenti Flash” a pagina 54.](#)

Nel seguente esempio vengono utilizzati tre componenti Button con i nomi di istanza a, b e c. Se si usano componenti diversi, assegnare ad essi un nome di istanza nella finestra di ispezione Proprietà e utilizzarli al punto 8.

2. Selezionare nella linea temporale un fotogramma nel quale, o prima del quale, è visualizzato il componente.
3. Aprire il pannello Azioni.
4. Aggiungere l'istruzione import seguente in modo da avere accesso alla funzione di costruzione per la creazione di una nuova dichiarazione di stile dalla classe CSSStyleDeclaration:

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

5. Utilizzare la sintassi seguente per creare un'istanza dell'oggetto CSSStyleDeclaration in modo da definire il nuovo formato di stile personalizzato:

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

6. Assegnare un nome alla dichiarazione di stile, ad esempio “myStyle”, nell'elenco `_global.styles` di dichiarazioni di stile personalizzato, quindi identificare l'oggetto contenente tutte le proprietà per la nuova dichiarazione di stile.

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

7. Per aggiungere proprietà all'oggetto `new_style`, utilizzare il metodo `setStyle()` ereditato dalla classe `UIObject`. Tali proprietà sono a loro volta associate alla dichiarazione di stile personalizzato `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. Nello stesso riquadro dello script, utilizzare la sintassi seguente per impostare la proprietà `styleName` di tre componenti specifici sul nome della dichiarazione di stile personalizzato:

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

È inoltre *possibile* accedere agli stili nella dichiarazione di stile personalizzato utilizzando i metodi `setStyle()` e `getStyle()` tramite la proprietà globale della dichiarazione `styleName`. Ad esempio, il codice seguente imposta lo stile `backgroundColor` nella dichiarazione di stile `myStyle`:

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

Tuttavia, dato che nei passaggi 5 e 6 è stata eseguita l'associazione dell'istanza `new_style` alla dichiarazione di stile, è possibile utilizzare una sintassi più sintetica, come `new_style.setStyle("themeColor", "haloOrange")`.

Per ulteriori informazioni sui metodi `setStyle()` e `getStyle()`, vedere `UIObject.setStyle()` e `UIObject.getStyle()`.

Impostazione di stili per una classe di componente

Per qualunque classe di componente (`Button`, `CheckBox` e così via) è possibile definire una dichiarazione di stile classe che imposti gli stili predefiniti di ogni istanza di tale classe. È necessario creare la dichiarazione di stile prima di creare le istanze. Per impostazione predefinita, alcuni componenti, quali `TextArea` e `TextInput`, dispongono di dichiarazioni di stile classe predefinite, in quanto le proprietà `borderStyle` e `backgroundColor` devono essere personalizzate.

ATTENZIONE

Se si sostituisce un foglio di stile classe, aggiungere al nuovo foglio di stile gli stili di quello precedente che si desidera mantenere; in caso contrario, vengono sovrascritti.

Con il codice seguente viene innanzitutto verificato se il tema corrente dispone già di una dichiarazione di stile per `CheckBox` e, in caso contrario, ne viene creata una. Il codice utilizza quindi il metodo `setStyle()` per definire una proprietà di stile per la dichiarazione di stile `CheckBox`; in questo caso “color” imposta su blu il colore di tutto il testo dell'etichetta della casella di controllo:

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

Per una tabella delle proprietà di stile che si possono impostare sul componente `CheckBox`, vedere “Uso degli stili con il componente `CheckBox`” nella *Guida di riferimento dei componenti*.

Le impostazioni di stile personalizzato hanno la priorità sulle impostazioni di stile ereditato e globale. Per un elenco relativo alla priorità degli stili, vedere [“Uso degli stili globale, personalizzato e classe nello stesso documento”](#) a pagina 97.

Impostazione di stili ereditari in un contenitore

Lo *stile ereditato* è uno stile che eredita il proprio valore dai componenti principali della gerarchia MovieClip del documento. Se lo stile di testo o di colore non è impostato a livello di istanza o classe o a livello personalizzato, il valore dello stile viene cercato automaticamente nella gerarchia MovieClip. Di conseguenza, gli stili impostati in un componente contenitore vengono ereditati dai componenti in esso contenuti.

Gli stili ereditari sono i seguenti:

- `fontFamily`
- `fontSize`
- `fontStyle`
- `fontWeight`
- `textAlign`
- `textIndent`
- Tutti gli stili di colore a valore singolo; ad esempio, `themeColor` è uno stile ereditario, a differenza di `alternatingRowColors`.

Il gestore di stili (`StyleManager`) indica a Flash se uno stile eredita il proprio valore da un altro stile. È inoltre possibile aggiungere degli stili in fase di runtime come stili ereditari. Per ulteriori informazioni, vedere [Classe `StyleManager`](#) nella *Guida di riferimento dei componenti*.

NOTA

Una delle principali differenze tra l'implementazione di stili per i componenti Flash e gli stili CSS applicati alle pagine HTML riguarda il valore `inherit` di CSS che non è supportato per i componenti Flash. Gli stili vengono o meno ereditati in base alla progettazione dei componenti.

Gli stili ereditari hanno la priorità sugli stili globali. Per un elenco relativo alla priorità degli stili, vedere [“Uso degli stili globale, personalizzato e classe nello stesso documento”](#) a pagina 97.

Nell'esempio seguente viene dimostrato come utilizzare gli stili ereditari con un componente Accordion disponibile in Flash Professional 8. La funzione degli stili ereditari è supportata in Flash Basic 8 e in Flash Professional 8.

Per creare un componente Accordion con stili ereditati dai componenti nei singoli riquadri Accordion:

1. Aprire un nuovo file FLA.
2. Trascinare un componente Accordion nello stage dal pannello Componenti.
3. Nella finestra di ispezione Proprietà, impostare il nome e le dimensioni del componente Accordion. In questo esempio, assegnare al componente il nome di istanza **accordion**.
4. Trascinare nella libreria un componente TextInput e un componente Button dal pannello Componenti.

I componenti trascinati nella libreria sono disponibili per lo script in fase di esecuzione.

5. Aggiungere al primo fotogramma della linea temporale il codice ActionScript seguente:

```
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);
```

Questo codice aggiunge due elementi secondari al componente Accordion e carica ogni elemento con un controllo TextInput e un controllo Button, utilizzati in questo esempio per illustrare l'ereditarietà degli stili.

6. Selezionare Controllo > Prova filmato per visualizzare il documento prima di aggiungere l'ereditarietà degli stili.
7. Aggiungere alla fine dello script nel primo fotogramma il codice ActionScript seguente:
`accordion.setStyle("fontStyle", "italic");`
8. Selezionare Controllo > Prova filmato per visualizzare le modifiche.

Si noti che l'impostazione `fontStyle` del componente Accordion non influisce solo sul testo di questo componente, ma anche sul testo associato ai componenti TextInput e Button in esso contenuti.

Uso degli stili globale, personalizzato e classe nello stesso documento

Se si definisce uno stile in una sola posizione all'interno di un documento, Flash utilizza tale definizione per rilevare il valore di una proprietà. Tuttavia, un singolo documento Flash può disporre di diverse impostazioni di stile, quali proprietà di stile impostate direttamente nelle istanze di componente, dichiarazioni di stile personalizzate, dichiarazioni di stile classe predefinite, stili ereditari e una dichiarazione di stile globale. In una situazione di questo tipo, Flash determina il valore di una proprietà cercandone la definizione in tutte queste posizioni secondo un ordine specifico.

Gli stili vengono cercati fino a quando non viene trovato un valore, secondo l'ordine seguente:

1. Viene cercata una proprietà di stile nell'istanza di componente.
2. La proprietà `styleName` dell'istanza viene esaminata per verificare se è stata assegnata ad essa una dichiarazione di stile personalizzata.
3. La proprietà viene cercata in una dichiarazione di stile classe predefinita.
4. Se lo stile è ereditario, viene cercato nella gerarchia principale un valore ereditato.
5. Viene quindi cercato lo stile nella dichiarazione di stile globale.
6. Se la proprietà non è ancora definita, essa assume il valore `undefined`.

Informazioni sulle proprietà dello stile di colore

Le proprietà dello stile di colore presentano un comportamento diverso rispetto alle proprietà di altri tipi. Tutte le proprietà relative ai colori hanno un nome che termina in "Color", ad esempio `backgroundColor`, `disabledColor` e `color`. Quando si modificano le proprietà dello stile, il colore viene immediatamente cambiato nell'istanza e in tutte le istanze secondarie appropriate. Tutte le altre modifiche delle proprietà di stile non fanno altro che contrassegnare l'oggetto per indicare che deve essere ridisegnato; le modifiche non vengono apportate fino al fotogramma successivo.

Il valore di una proprietà di stile del colore può essere un numero, una stringa o un oggetto. Se è un numero, rappresenta il valore RGB del colore in formato numerico esadecimale (0xRRGGBB). Se il valore è una stringa, deve essere il nome di un colore.

I nomi di colore sono stringhe associate ai colori usati più comunemente. È possibile aggiungere nuovi nomi di colore utilizzando il gestore di stili (vedere [Classe StyleManager](#) nella *Guida di riferimento dei componenti*). L'elenco seguente riporta i nomi di colore predefiniti:

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

NOTA

Se il nome del colore non viene definito, il componente potrebbe non risultare corretto.

Per creare nomi di colore personalizzati, è possibile utilizzare qualunque identificatore di ActionScript valido, come ad esempio "WindowText" o "ButtonText". Per definire colori nuovi, utilizzare il gestore di stili (StyleManager), come indicato di seguito:

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

La maggior parte dei componenti non può gestire un oggetto sotto forma di valore di una proprietà dello stile di colore. Tuttavia, alcuni componenti possono gestire oggetti colore che rappresentano sfumature o altre combinazioni di colori. Per ulteriori informazioni, consultare la sezione "Uso degli stili" della voce relativa a ciascun componente nella *Guida di riferimento dei componenti*.

Unitamente ai nomi dei colori, le dichiarazioni di stile classe consentono di controllare agevolmente i colori del testo e dei simboli nelle schermate. Ad esempio, per creare una schermata di configurazione iniziale che somigli a quella di Microsoft Windows, è necessario definire nomi di colore come `ButtonText` e `WindowText` e dichiarazioni di stile di classe come `Button`, `CheckBox` e `Window`.

NOTA

Le proprietà dello stile di alcuni componenti sono array di colori, come `alternatingRowColors`. Questi stili devono essere impostati solo come array di valori RGB numerici anziché come nomi di colori.

Personalizzazione delle animazioni dei componenti

Molti componenti, quali `Accordion`, `ComboBox` e `Tree`, utilizzano un'animazione per visualizzare il passaggio tra i diversi stati del componente, ad esempio quando si scorrono gli elementi secondari di `Accordion`, si espande l'elenco a discesa di `ComboBox` e si espandono o si comprimono le cartelle di `Tree`. L'animazione viene utilizzata dai componenti anche quando si seleziona e deselecta un elemento, ad esempio una riga di un elenco.

È possibile controllare l'aspetto di queste animazioni utilizzando i seguenti stili:

Stile dell'animazione	Descrizione
<code>openDuration</code>	La durata in millisecondi della transizione per l'andamento aperto nei componenti <code>Accordion</code> , <code>ComboBox</code> e <code>Tree</code> . Il valore predefinito è 250.
<code>openEasing</code>	Un riferimento a una funzione di interpolazione che controlla l'animazione dello stato nei componenti <code>Accordion</code> , <code>ComboBox</code> e <code>Tree</code> . L'equazione predefinita utilizza una formula <code>sine in/out</code> .
<code>popupDuration</code>	La durata in millisecondi della transizione all'apertura del menu nel componente <code>Menu</code> . Il valore predefinito è 150. Si noti, tuttavia, che l'animazione utilizza sempre l'equazione predefinita <code>sine in/out</code> .
<code>selectionDuration</code>	La durata in millisecondi della transizione nei componenti <code>ComboBox</code> , <code>DataGrid</code> , <code>List</code> e <code>Tree</code> dallo stato normale allo stato selezionato o viceversa. Il valore predefinito è 200.
<code>selectionEasing</code>	Un riferimento a una funzione di interpolazione che controlla l'animazione della selezione nei componenti <code>ComboBox</code> , <code>DataGrid</code> , <code>List</code> e <code>Tree</code> . Questo stile è valido solo per la transizione da uno stato normale a uno stato selezionato. L'equazione predefinita utilizza una formula <code>sine in/out</code> .

Il pacchetto `mx.transitions.easing` fornisce sei classi per il controllo dell'andamento:

Classe di andamento	Descrizione
Back	Si estende oltre l'intervallo di transizione a una o entrambe le estremità per fornire un leggero effetto di "sconfinamento".
Bounce	Fornisce un effetto di rimbalzo interamente all'interno dell'intervallo di transizione a una o entrambe le estremità. Il numero di rimbalzi dipende dalla durata: maggiore è la durata, maggiore è il numero di rimbalzi.
Elastic	Fornisce un effetto elastico non incluso nell'intervallo di transizione a una o entrambe le estremità. La quantità di elasticità non dipende dalla durata.
None	Fornisce un movimento normale dall'inizio alla fine, senza effetti, rallentamento o accelerazione. Questa transizione è comunemente nota come <i>transizione lineare</i> .
Regular	Fornisce un movimento più lento a una o entrambe le estremità per ottenere un effetto di accelerazione o di rallentamento o entrambi.
Strong	Fornisce un movimento molto rallentato a una o entrambe le estremità. Questo effetto è simile a quello fornito da Regular, ma più pronunciato.

Ogni classe del pacchetto `mx.transitions.easing` dispone dei tre metodi di andamento seguenti:

Metodo di andamento	Descrizione
<code>easeIn</code>	Fornisce l'effetto di andamento all'inizio della transizione.
<code>easeOut</code>	Fornisce l'effetto di andamento alla fine della transizione.
<code>easeInOut</code>	Fornisce l'effetto di andamento all'inizio e alla fine della transizione.

Poiché i metodi di andamento sono metodi statici delle classi di andamento, non è necessario creare un'istanza delle classi di andamento. I metodi vengono utilizzati nelle chiamate a `setStyle()`, come nell'esempio seguente.

```
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

L'equazione predefinita utilizzata da tutte le transizioni non è disponibile nelle classi di andamento sopra elencate. Per specificare l'uso del metodo di andamento predefinito in un componente dopo che è stato specificato un altro metodo di andamento, chiamare `setStyle("openEasing", null)`.

Per ulteriori informazioni, vedere [“Applicazione dei metodi di andamento ai componenti”](#) nella *Guida di riferimento dei componenti*.

Recupero dei valori delle proprietà di stile

Per recuperare il valore di una proprietà di stile, utilizzare `UIObject.getStyle()`. Tutti i componenti sottoclasse di `UIObject`, ovvero tutti i componenti della versione 2 ad eccezione dei componenti Media, ereditano il metodo `getStyle()`. Questo significa che è possibile chiamare `getStyle()` da qualsiasi istanza di componente, esattamente come `setStyle()`.

Il codice seguente ottiene il valore dello stile `themeColor` e lo assegna alla variabile `oldStyle`:

```
var myCheckBox:mx.controls.CheckBox;  
var oldFontSize:Number  
  
oldFontSize = myCheckBox.getStyle("fontSize");  
trace(oldFontSize);
```

Informazioni sull'associazione di skin ai componenti

Gli skin sono i simboli di clip filmato utilizzati da un componente per visualizzare il proprio aspetto. La maggior parte contiene forme che rappresentano l'aspetto del componente, mentre alcuni contengono soltanto il codice `ActionScript` che permette di disegnare il componente nel documento.

I componenti della versione 2 sono clip compilati e non è quindi possibile vederne le risorse nella libreria. Tuttavia, la versione installata di Flash include file FLA contenenti tutti gli skin dei componenti; i file di questo tipo sono chiamati *temi*. Ogni tema ha aspetto e comportamento diversi dagli altri, ma contiene skin con nomi di simbolo e identificatori di concatenamento uguali. Questo consente di trascinare un tema all'interno di un documento sullo stage per cambiarne l'aspetto. I file FLA dei temi vengono usati anche per modificare gli skin dei componenti. Gli skin si trovano nella cartella Themes del pannello Libreria di ogni file FLA di un tema. Per ulteriori informazioni sui temi, vedere [“Informazioni sui temi”](#) a pagina 115.

Ogni componente è composto da molti skin. Ad esempio, la freccia giù del sottocomponente `ScrollBar` è composta da quattro skin: `ScrollDownArrowDisabled`, `ScrollDownArrowDown`, `ScrollDownArrowOver` e `ScrollDownArrowUp`. Nell'intero sottocomponente `ScrollBar` vengono utilizzati 13 simboli di skin diversi.

Alcuni componenti condividono gli skin: ad esempio, i componenti che utilizzano le barre di scorrimento, quali ComboBox, List e ScrollPane, condividono gli skin nella cartella ScrollBar Skins. Per modificare l'aspetto dei componenti, è possibile sia modificare gli skin esistenti che creare skin nuovi.

Il file AS che definisce ciascuna classe di componente contiene il codice che consente di caricare skin specifici per il componente. Ogni skin di componente corrisponde a una proprietà che viene assegnata all'identificatore di concatenamento di un simbolo di skin. Ad esempio, lo stato giù (down) della freccia giù del componente ScrollBar ha come nome di proprietà di skin `downArrowDownName`. Il valore predefinito della proprietà `downArrowDownName` è "ScrollDownArrowDown", che è anche l'identificatore di concatenamento del simbolo di skin nel file FLA del tema. È possibile modificare gli skin esistenti e applicarli a tutti i componenti che li utilizzano modificando il simbolo di skin e mantenendo l'identificatore di concatenamento esistente. È possibile creare nuovi skin e applicarli a istanze di componente specifiche impostando le proprietà degli skin per le singole istanze desiderate. Per modificare le proprietà degli skin non è necessario modificare il file AS del componente, in quanto è possibile passarne i valori alla funzione di costruzione del componente quando questo viene creato nel documento.

Le proprietà degli skin di ogni componente sono elencate nella voce relativa al componente nel Dizionario dei componenti. Ad esempio, le proprietà degli skin del componente Button si trovano in Guida di riferimento dei componenti > Componente Button > Personalizzazione del componente Button > Uso degli skin con il componente Button.

Per associare gli skin a un componente in base alle proprie esigenze, scegliere uno dei metodi indicati di seguito. I metodi sono elencati in ordine crescente di difficoltà.

- Modificare gli skin associati a tutte le istanze di un componente specifico in un singolo documento, copiando e modificando i singoli elementi skin. Vedere [“Modifica degli skin dei componenti di un documento” a pagina 103](#).
Questo metodo di associazione degli skin è consigliato per gli utenti meno esperti, in quanto non richiede la creazione di script.
- Sostituire tutti gli skin di un documento con una nuova serie di skin, per ogni tipo di componente che condivide lo stesso aspetto, applicando un tema. Vedere [“Informazioni sui temi” a pagina 115](#).
Questo metodo di associazione degli skin è consigliato per garantire un aspetto uniforme in tutti i componenti e in documenti diversi.
- Collegare il colore di un elemento skin a una proprietà di stile, aggiungendo codice ActionScript allo skin per registrarlo come elemento skin colorato. Vedere [“Collegamento del colore degli skin agli stili” a pagina 106](#).

- Utilizzare skin diversi per più istanze dello stesso componente, creando nuovi skin e impostando le proprietà degli skin. Vedere [“Creazione di nuovi skin dei componenti” a pagina 105](#) e [“Applicazione di nuovi skin a un componente” a pagina 108](#).
- Modificare gli skin in un sottocomponente, ad esempio una barra di scorrimento in un componente List, creando una sottoclasse per il componente. Vedere [“Applicazione di nuovi skin a un sottocomponente” a pagina 109](#).
- Modificare gli skin di un sottocomponente che non sono direttamente accessibili dal componente principale, ad esempio un componente List in un componente ComboBox, sostituendo le proprietà di skin nel prototipo. Vedere [“Modifica delle proprietà di skin in un sottocomponente” a pagina 113](#).

Modifica degli skin dei componenti di un documento

Per modificare gli skin associati a tutte le istanze di un componente specifico presenti in un singolo documento, copiare nel documento i simboli di skin del tema e modificarne la grafica nel modo desiderato.

La procedura riportata di seguito è molto simile a quella per la creazione e l'applicazione di un nuovo tema (vedere [“Informazioni sui temi” a pagina 115](#)). La principale differenza è che questa procedura illustra come copiare i simboli in un singolo documento direttamente dal tema già in uso e come modificare solo un numero ridotto di tutti gli skin disponibili. Questa procedura è quindi appropriata quando tutte le modifiche vengono apportate a un unico documento e gli skin vengono modificati solo per pochi componenti. In caso di modifica di skin condivisi tra più documenti o riferiti a numerosi componenti, potrebbe essere più semplice modificare gli skin se si crea un nuovo tema.

Un articolo sulle funzioni avanzate dell'associazione di skin è disponibile in linea nel Centro per sviluppatori Macromedia all'indirizzo www.macromedia.com/devnet/mx/flash/articles/skinning_2004.html.

Per modificare gli skin dei componenti di un documento:

1. Se il tema Sample è già stato applicato a un documento, passare al punto 5.
2. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file SampleTheme.fla.

Questo file si trova nella cartella Configuration dell'applicazione. Per conoscere l'esatta posizione nel proprio sistema operativo, vedere [“Informazioni sui temi” a pagina 115](#).

3. Nel pannello Libreria del tema, selezionare Flash UI Components 2/Themes/MMDefault, quindi trascinare la cartella Assets di qualsiasi componente del documento nella libreria del documento stesso.

Ad esempio, trascinare nella libreria ThemeApply.fla la cartella RadioButton Assets.

4. Se nella libreria sono state trascinate le cartelle Assets dei singoli componenti, assicurarsi che il simbolo Assets di ogni componente sia impostato su Esporta nel primo fotogramma.

Ad esempio, la cartella Assets del componente RadioButton è denominata RadioButton Assets e contiene un simbolo denominato RadioButtonAssets che a sua volta contiene tutti i simboli delle singole risorse. Se si imposta Esporta nel primo fotogramma per il simbolo RadioButtonAssets, vengono esportati nel primo fotogramma anche tutti i simboli delle singole risorse.

5. Fare doppio clic sul simbolo di uno skin per aprirlo in modalità di modifica del simbolo.

Ad esempio, aprire il simbolo States/RadioFalseDisabled.

6. Modificare il simbolo oppure eliminare la grafica e crearne una nuova.

Per aumentare il livello di ingrandimento, selezionare Visualizza > Ingrandisci. Quando si modifica uno skin, è necessario mantenerne il punto di registrazione per assicurare una visualizzazione corretta. L'angolo superiore sinistro di tutti i simboli modificati deve essere impostato su (0,0).

Ad esempio, assegnare al cerchio interno il colore grigio chiaro.

7. Al termine della modifica del simbolo di skin, fare clic sul pulsante Indietro, sul lato sinistro della barra delle informazioni nella parte superiore dello stage, per tornare alla modalità di modifica del documento.

8. Ripetere le operazioni elencate ai punti da 5 a 7 finché non sono stati modificati tutti gli skin desiderati.

NOTA

L'anteprima dal vivo dei componenti sullo stage non riflette le modifiche degli skin.

9. Selezionare Controllo > Prova filmato.

In questo esempio, assicurarsi che lo stage contenga un'istanza RadioButton e impostarne la proprietà `enabled` su `false` nel pannello Azioni per visualizzare il nuovo aspetto disabilitato di RadioButton.

Creazione di nuovi skin dei componenti

Se si desidera utilizzare uno skin specifico per un'istanza di un componente e uno skin diverso per un'altra istanza dello stesso componente, aprire il file FLA di un tema e creare un nuovo simbolo di skin. I componenti sono progettati per facilitare l'uso di skin diversi per istanze diverse.

Per creare un nuovo skin:

1. Selezionare File > Apri e aprire il file FLA del tema da utilizzare come modello.
2. Selezionare File > Salva con nome e assegnare al file un nome univoco, ad esempio **MyTheme.flc**.
3. Selezionare gli skin da modificare, in questo esempio RadioTrueUp.
Gli skin si trovano nella cartella Themes/MMDefault/*Componente Assets*, in questo esempio Themes/MMDefault/RadioButton Assets/States.
4. Selezionare Duplica dal menu Opzioni della libreria o dal menu visualizzato facendo clic con il pulsante destro del mouse sul simbolo, quindi assegnare un nome univoco al simbolo, ad esempio **MyRadioTrueUp**.
5. Fare clic su Avanzato nella finestra di dialogo Proprietà simbolo, quindi selezionare Esporta per ActionScript.
Viene immesso automaticamente un identificatore di concatenamento che corrisponde al nome del simbolo.
6. Fare doppio clic sul nuovo skin nella libreria per aprirlo in modalità di modifica del simbolo.
7. Modificare il clip filmato o eliminarlo e crearne uno nuovo.
Per aumentare il livello di ingrandimento, selezionare Visualizza > Ingrandisci. Quando si modifica uno skin, è necessario mantenerne il punto di registrazione per assicurare una visualizzazione corretta. L'angolo superiore sinistro di tutti i simboli modificati deve essere impostato su (0,0).
8. Al termine della modifica del simbolo di skin, fare clic sul pulsante Indietro, sul lato sinistro della barra delle informazioni nella parte superiore dello stage, per tornare alla modalità di modifica del documento.
9. Selezionare File > Salva ma non chiudere il file MyTheme.flc. È necessario creare un nuovo documento nel quale applicare lo skin modificato a un componente.

Per ulteriori informazioni, vedere [“Applicazione di nuovi skin a un componente”](#) a pagina 108, [“Applicazione di nuovi skin a un sottocomponente”](#) a pagina 109 o [“Modifica delle proprietà di skin in un sottocomponente”](#) a pagina 113.

NOTA

Le modifiche apportate agli skin dei componenti non vengono mostrate in Flash quando si visualizzano i componenti nello stage con la funzione Anteprima dal vivo.

Collegamento del colore degli skin agli stili

L'architettura dei componenti della versione 2 facilita il collegamento di una risorsa visiva di un elemento skin a uno stile impostato nel componente che utilizza lo skin. Per registrare un'istanza di clip filmato o un intero elemento skin in uno stile, aggiungere il codice ActionScript nella linea temporale dello skin per chiamare

```
mx.skins.ColoredSkinElement.setColorStyle(targetMovieClip, styleName).
```

Per collegare uno skin a una proprietà di stile:

1. Se il tema Sample è già stato applicato a un documento, passare al punto 5.
2. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file SampleTheme fla.

Questo file si trova nella cartella Configuration dell'applicazione. Per conoscere l'esatta posizione nel proprio sistema operativo, vedere [“Informazioni sui temi”](#) a pagina 115.

3. Nel pannello Libreria del tema, selezionare Flash UI Components 2/Themes/MMDefault, quindi trascinare la cartella Assets di qualsiasi componente del documento nella libreria del documento stesso.

Ad esempio, trascinare nella libreria di destinazione la cartella RadioButton Assets.

4. Se sono state trascinate nella libreria le cartelle Assets dei singoli componenti, assicurarsi che il simbolo Assets di ogni componente sia impostato su Esporta nel primo fotogramma.

Ad esempio, la cartella Assets del componente RadioButton è denominata RadioButton Assets e contiene un simbolo denominato RadioButtonAssets che a sua volta contiene tutti i simboli delle singole risorse. Se si imposta Esporta nel primo fotogramma per il simbolo RadioButtonAssets, vengono esportati nel primo fotogramma anche tutti i simboli delle singole risorse.

5. Fare doppio clic sul simbolo di uno skin per aprirlo in modalità di modifica del simbolo. Ad esempio, aprire il simbolo States/RadioFalseDisabled.
6. Se l'elemento da colorare è un simbolo grafico e non un'istanza di clip filmato, utilizzare **Elabora > Converti in simbolo** per convertirlo in un'istanza di clip filmato.

In questo esempio, convertire l'immagine grafica centrale, vale a dire un'istanza del simbolo grafico `RadioShape1`, in un simbolo di clip filmato, quindi assegnare ad esso il nome **Inner Circle**. Non è necessario selezionare **Esporta** per **ActionScript**.

È buona norma, anche se non necessario, spostare il simbolo di clip filmato appena creato nella cartella **Elements** delle risorse di componente in fase di modifica.

7. Se nel punto precedente è stato convertito un simbolo grafico in un'istanza di clip filmato, assegnare un nome all'istanza in modo da poterla impostare come destinazione in **ActionScript**.

In questo esempio, immettere il nome di istanza **innerCircle**.

8. Aggiungere il codice **ActionScript** per registrare l'elemento skin o un'istanza di clip filmato in esso contenuta come elemento skin colorato.

Ad esempio, aggiungere il codice seguente alla linea temporale dell'elemento skin.

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

In questo esempio viene utilizzato un colore corrispondente a un nome di stile esistente nello stile **Sample**. Se possibile, è preferibile utilizzare nomi di stile corrispondenti agli standard **CSS** (**Cascading Style Sheet**) ufficiali oppure gli stili forniti nei temi **Halo** e **Sample**.

9. Ripetere le operazioni elencate ai punti da 5 a 8 finché non sono stati modificati tutti gli skin desiderati.

In questo esempio, ripetere questi punti per lo skin **RadioTrueDisabled**: in questo caso, non convertire l'elemento grafico esistente in un clip filmato, ma eliminarlo e trascinare nell'elemento skin **RadioTrueDisabled** il simbolo esistente **Inner Circle**.

10. Al termine della modifica del simbolo di skin, fare clic sul pulsante **Indietro**, sul lato sinistro della barra delle informazioni nella parte superiore dello stage, per tornare alla modalità di modifica del documento.

11. Trascinare nello stage un'istanza del componente.

In questo esempio, trascinare nello stage due componenti **RadioButton**, impostarne uno come selezionato e utilizzare **ActionScript** per impostare entrambi come disattivati e vedere le modifiche.

12. Aggiungere il codice **ActionScript** al documento per impostare la nuova proprietà di stile nelle istanze del componente o a livello globale.

In questo esempio, impostare la proprietà a livello globale, come indicato di seguito:

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

13. Selezionare **Controllo > Prova filmato**.

Applicazione di nuovi skin a un componente

Dopo aver creato uno skin, occorre applicarlo a un componente in un documento. È possibile utilizzare il metodo `createClassObject()` per la creazione dinamica delle istanze di componente oppure posizionare manualmente le istanze di componente sullo stage. A seconda del metodo utilizzato per aggiungere i componenti a un documento, esistono due modi diversi di applicare gli skin alle istanze di componente.

Per creare dinamicamente un componente e applicare un nuovo skin:

1. Selezionare File > Nuovo per creare un nuovo documento Flash.
2. Selezionare File > Salva e assegnare al file un nome univoco, ad esempio **DynamicSkinning.fla**.
3. Trascinare dei componenti nella libreria dal pannello Componenti, compreso il componente di cui è stato modificato lo skin, in questo esempio **RadioButton**.
In questo modo i simboli vengono aggiunti alla libreria del documento, senza essere resi visibili nel documento.
4. Trascinare **MyRadioTrueUp** ed eventuali altri simboli personalizzati da **MyTheme.fla** alla libreria di **DynamicSkinning.fla**.
In questo modo i simboli vengono aggiunti alla libreria del documento, senza essere resi visibili nel documento.
5. Aprire il pannello Azioni e immettere il codice seguente per il fotogramma 1:

```
import mx.controls.RadioButton;
createClassObject(RadioButton, "myRadio", 0,
    {trueUpIcon:"MyRadioTrueUp", label: "My Radio Button"});
```
6. Selezionare Controllo > Prova filmato.

Per aggiungere manualmente un componente allo stage e applicare un nuovo skin:

1. Selezionare File > Nuovo per creare un nuovo documento Flash.
2. Selezionare File > Salva e assegnare al file un nome univoco, ad esempio **ManualSkinning.fla**.
3. Trascinare dei componenti nello stage dal pannello Componenti, compreso il componente di cui è stato modificato lo skin, in questo esempio, **RadioButton**.
4. Trascinare **MyRadioTrueUp** ed eventuali altri simboli personalizzati da **MyTheme.fla** alla libreria di **ManualSkinning.fla**.
In questo modo i simboli vengono aggiunti alla libreria del documento, senza essere resi visibili nel documento.

5. Selezionare il componente RadioButton sullo stage e aprire il pannello Azioni.
6. Associare il codice seguente all'istanza di RadioButton:

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

7. Selezionare Controllo > Prova filmato.

Applicazione di nuovi skin a un sottocomponente

In alcune situazioni può rendersi necessario modificare gli skin di un sottocomponente in un componente, ma le proprietà interessate non sono direttamente disponibili; ad esempio, non esiste un modo per modificare direttamente gli skin della barra di scorrimento in un componente List. Il codice riportato di seguito consente di accedere agli skin delle barre di scorrimento. Tutte le barre di scorrimento create dopo l'esecuzione di questo codice dispongono a loro volta degli skin nuovi.

Se un componente è composto da sottocomponenti, questi vengono indicati nella voce relativa al componente nella *Guida di riferimento dei componenti*.

Per applicare un nuovo skin a un sottocomponente:

1. Seguire le procedure riportate in [“Creazione di nuovi skin dei componenti” a pagina 105](#), modificando però lo skin di una barra di scorrimento. In questo esempio, modificare lo skin ScrollDownArrowDown e assegnare ad esso il nuovo nome **MyScrollDownArrowDown**.
2. Selezionare File > Nuovo per creare un nuovo documento Flash.
3. Selezionare File > Salva e assegnare al file un nome univoco, come **SubcomponentProject.fla**.
4. Trascinare nella libreria il componente List del pannello Componenti.
In questo modo il componente viene aggiunto al pannello Libreria, senza essere reso visibile nel documento.
5. Trascinare MyScrollDownArrowDown ed eventuali altri simboli personalizzati da MyTheme.fla alla libreria di SubcomponentProject.fla.
In questo modo il simbolo viene aggiunto al pannello Libreria, senza essere reso visibile nel documento.

6. Effettuare una delle seguenti operazioni:

- Per modificare tutte le barre di scorrimento in un documento, immettere il codice seguente nel pannello Azioni sul fotogramma 1 della linea temporale:

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

È quindi possibile immettere il codice seguente nel fotogramma 1 per creare un elenco in modo dinamico:

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

Oppure è possibile trascinare un componente List nello stage dalla libreria.

- Per modificare una barra di scorrimento specifica in un documento, immettere il codice seguente nel pannello Azioni sul fotogramma 1 della linea temporale:

```
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

Impostare una quantità sufficiente di dati affinché le barre di scorrimento vengano visualizzate oppure impostare la proprietà `vScrollPolicy` su `true`.

7. Selezionare Controllo > Prova filmato.

È possibile inoltre impostare skin di sottocomponente per tutti i componenti in un documento mediante l'impostazione della proprietà `skin` sull'oggetto `prototype` del sottocomponente nella sezione `#initclip` di un simbolo di skin.

Per utilizzare `#initclip` in modo da applicare uno skin modificato a tutti i componenti di un documento:

1. Seguire le procedure riportate in [“Creazione di nuovi skin dei componenti”](#) a pagina 105, modificando però lo skin di una barra di scorrimento. In questo esempio, modificare lo skin `ScrollDownArrowDown` e assegnargli il nuovo nome `MyScrollDownArrowDown`.
2. Selezionare File > Nuovo e creare un nuovo documento Flash. Salvare il documento con un nome univoco, ad esempio `SkinsInitExample.fla`.
3. Selezionare il simbolo `MyScrollDownArrowDown` dalla libreria di esempio dei temi modificati e trascinarlo nella libreria di `SkinsInitExample.fla`.

Questa operazione aggiunge il simbolo alla libreria senza renderlo visibile nello stage.

4. Selezionare MyScrollDownArrowDown nella libreria SkinsInitExample fla e selezionare Concatenamento dal menu Opzioni della libreria.
5. Selezionare la casella di controllo Esporta per ActionScript. Fare clic su OK.
L'opzione Esporta nel primo fotogramma dovrebbe già essere selezionata; in caso contrario, selezionarla.
6. Fare doppio clic su MyScrollDownArrowDown nella libreria per aprirlo nella modalità di modifica del simbolo.
7. Immettere il codice seguente sul fotogramma 1 del simbolo MyScrollDownArrowDown:

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

8. Per aggiungere un componente List al documento, effettuare una delle operazioni seguenti:

- Trascinare un componente List nello stage dal pannello Componenti. Immettere parametri di etichetta sufficienti per fare in modo che la barra di scorrimento verticale venga visualizzata.
- Trascinare nella libreria un componente List dal pannello Componenti. Immettere il codice seguente sul fotogramma 1 della linea temporale principale di SkinsInitExample fla:

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

NOTA

Aggiungere una quantità di dati sufficiente affinché la barra di scorrimento verticale venga visualizzata oppure impostare `vScrollPolicy` su `true`.

Nell'esempio seguente viene spiegato come creare lo skin di elementi che si trovano già nello stage. In questo esempio vengono creati solo gli skin del componente List e non quelli dei componenti TextArea o ScrollPane.

Per utilizzare #initclip in modo da applicare uno skin modificato a componenti specifici di un documento:

1. Seguire le procedure riportate in [“Modifica degli skin dei componenti di un documento” a pagina 103](#), modificando però lo skin di una barra di scorrimento. In questo esempio, modificare lo skin ScrollDownArrowDown e assegnargli il nuovo nome **MyScrollDownArrowDown**.
2. Selezionare File > Nuovo e creare un nuovo documento Flash.
3. Selezionare File > Salva e assegnare al file un nome univoco, ad esempio **MyComboTest fla**.

4. Trascinare MyScrollDownArrowDown dalla libreria dei temi alla libreria MyVScrollTest.fla.
5. Selezionare Inserisci > Nuovo simbolo e assegnare al simbolo un nome univoco, ad esempio **MyVScrollBar**.
6. Selezionare la casella di controllo Esporta per ActionScript. Fare clic su OK.
L'opzione Esporta nel primo fotogramma dovrebbe già essere selezionata; in caso contrario, selezionarla.
7. Immettere il codice seguente sul fotogramma 1 del simbolo MyVScrollBar:

```
#initclip 10
import MyVScrollBar
Object.registerClass("VScrollBar", MyVScrollBar);
#endinitclip
```
8. Trascinare un componente List dal pannello Componenti allo stage.
9. Nella finestra di ispezione Proprietà, immettere il numero di parametri Label necessario per visualizzare la barra di scorrimento verticale.
10. Selezionare File > Salva.
11. Selezionare File > Nuovo e creare un nuovo file ActionScript.
12. Immettere il codice seguente:

```
import mx.controls.VScrollBar
import mx.controls.List
class MyVScrollBar extends VScrollBar{
    function init():Void{
        if (_parent instanceof List){
            downArrowDownName = "MyScrollDownArrowDown";
        }
        super.init();
    }
}
```
13. Selezionare File > Salva e salvare il file con il nome **MyVScrollBar.as**.
14. Fare clic su un'area vuota nello stage e, nella finestra di ispezione Proprietà, fare clic sul pulsante Impostazioni pubblicazione.
15. Fare clic sul pulsante Impostazioni a destra della casella di riepilogo Versione di ActionScript.
16. Fare clic sul pulsante Aggiungi nuovo percorso (+) per aggiungere un nuovo percorso di classe e selezionare il pulsante Destinazione per individuare la posizione del file MyVScrollBar.as sul disco rigido.
17. Selezionare Controllo > Prova filmato.

Modifica delle proprietà di skin in un sottocomponente

Se un componente non supporta direttamente le variabili di skin, è possibile creare una sottoclasse del componente e sostituirne gli skin. Ad esempio, il componente ComboBox non supporta direttamente l'assegnazione di skin all'elenco a discesa, in quanto come elenco a discesa utilizza un componente List.

Se un componente è composto da sottocomponenti, questi vengono indicati nella voce relativa al componente nella *Guida di riferimento dei componenti*.

Per assegnare lo skin a un sottocomponente:

1. Seguire le procedure riportate in [“Modifica degli skin dei componenti di un documento” a pagina 103](#), modificando però lo skin di una barra di scorrimento. In questo esempio, modificare lo skin ScrollDownArrowDown e assegnargli il nuovo nome **MyScrollDownArrowDown**.

2. Selezionare File > Nuovo e creare un nuovo documento Flash.

3. Selezionare File > Salva e assegnare al file un nome univoco, come ad esempio **MyComboTest.fla**.

4. Trascinare MyScrollDownArrowDown dalla libreria dei temi precedente alla libreria di MyComboTest.fla.

In questo modo il simbolo viene aggiunto alla libreria, senza essere reso visibile nello stage.

5. Selezionare Inserisci > Nuovo simbolo e assegnare al simbolo un nome univoco, ad esempio **MyComboBox**.

6. Selezionare la casella di controllo Esporta per ActionScript e fare clic su OK.

L'opzione Esporta nel primo fotogramma dovrebbe già essere selezionata; in caso contrario, selezionarla.

7. Nel pannello Azioni, immettere il codice seguente nel fotogramma 1 del simbolo MyComboBox:

```
#initclip 10
import MyComboBox
Object.registerClass("ComboBox", MyComboBox);
#endinitclip
```

8. Al termine della modifica del simbolo, fare clic sul pulsante Indietro, sul lato sinistro della barra delle informazioni nella parte superiore dello stage, per tornare alla modalità di modifica del documento.
9. Trascinare un componente ComboBox nello stage.

10. Nella finestra di ispezione Proprietà, immettere il numero di parametri Label necessario per visualizzare la barra di scorrimento verticale.

11. Selezionare File > Salva.

12. Selezionare File > Nuovo e creare un nuovo file **ActionScript**.

13. Immettere il codice seguente:

```
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. Selezionare File > Salva e salvare il file con il nome **MyComboBox.as**.

15. Tornare al file **MyComboTest fla**.

16. Fare clic su un'area vuota nello stage e, nella finestra di ispezione Proprietà, fare clic sul pulsante Impostazioni pubblicazione.

17. Fare clic sul pulsante Impostazioni a destra della casella di riepilogo Versione di **ActionScript**.

18. Fare clic sul pulsante Aggiungi nuovo percorso (+) per aggiungere un nuovo percorso di classe e selezionare il pulsante Destinazione per individuare la posizione del file **MyComboBox.as** sul disco rigido.

19. Selezionare Controllo > Prova filmato.

Informazioni sui temi

I temi sono raccolte di stili e skin. Il tema predefinito di Flash è Halo (HaloTheme fla), un tema che consente di garantire agli utenti un'esperienza stimolante e ricca di espressività. In Flash sono inclusi temi aggiuntivi, ad esempio Sample (SampleTheme fla). Questo tema è un esempio di come sia possibile utilizzare più stili per la personalizzazione, in quanto utilizza molti più stili di quelli inclusi nel tema Halo. In un'installazione predefinita, i file dei temi si trovano nelle cartelle seguenti:

- In Windows: C:\Programmi\Macromedia\Flash 8\lingua\Configuration\ComponentsFLA\
- In Macintosh: HD/Applicazioni/Macromedia Flash 8/Configuration/ComponentFLA/

È possibile creare temi nuovi e applicarli a un'applicazione per modificare l'aspetto di tutti i componenti. Ad esempio, è possibile creare temi che imitano l'aspetto del sistema operativo nativo.

I componenti utilizzano skin, ovvero simboli grafici o di clip filmato, per visualizzare il proprio aspetto. Il file AS che definisce ciascun componente contiene il codice che consente di caricare skin specifici per il componente. È possibile creare facilmente un tema nuovo facendo una copia del tema Halo o Sample e modificandone la grafica negli skin.

Un tema può anche contenere i valori predefiniti di una nuova serie di stili. Per creare una dichiarazione di stile `_global` ed eventuali altre dichiarazioni di stile, scrivere istruzioni ActionScript. Per ulteriori informazioni, vedere [“Modifica dei valori predefiniti delle proprietà di stile di un tema” a pagina 119](#).

Passaggio tra temi

Con Macromedia Flash vengono installati due temi: Halo e Sample. Le informazioni di riferimento per ogni componente contengono una tabella delle proprietà di stile che si possono impostare per uno o entrambi i temi. Quando si visualizza una tabella delle proprietà di stile, come quella del componente Button in “Uso degli stili con il componente Button” nella *Guida di riferimento dei componenti*, osservare quale tema supporta lo stile desiderato: "Halo", "Sample" o "Entrambi" (indica che entrambi i temi supportano quella proprietà di stile).

Halo è il tema predefinito per i componenti. Se si desidera utilizzare il tema Sample, è quindi necessario passare dal tema corrente Halo a Sample.

Per passare al tema Sample:

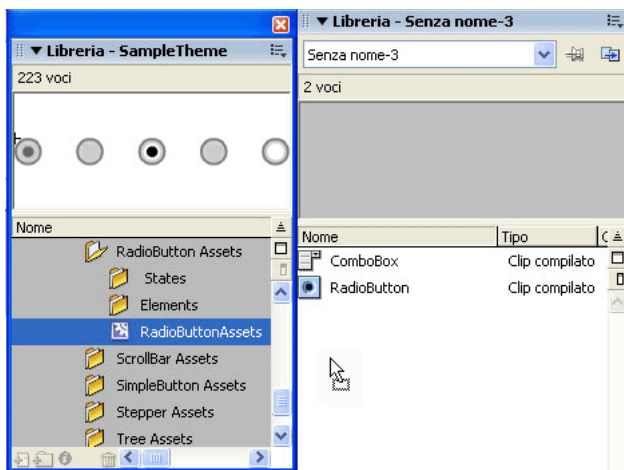
1. Selezionare File > Apri, quindi aprire il documento che utilizza i componenti della versione 2 in Flash oppure selezionare File > Nuovo e creare un documento nuovo che utilizzi i componenti della versione 2.

2. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file SampleTheme.fla da applicare al documento.

Questo file si trova nella cartella Configuration dell'applicazione. Per conoscere l'esatta posizione nel proprio sistema operativo, vedere [“Informazioni sui temi” a pagina 115](#).

3. Nel pannello Libreria del tema SampleTheme.fla, selezionare Flash UI Components 2/ Themes/MMDefault, quindi trascinare le cartelle Assets di qualsiasi componente del documento nel pannello Libreria del documento Flash.

Ad esempio, trascinare nella libreria la cartella RadioButton Assets.



Se non si sa esattamente quali componenti si trovino nel documento, trascinare nello stage l'intero clip filmato del tema Sample. Gli skin vengono assegnati automaticamente ai componenti del documento.

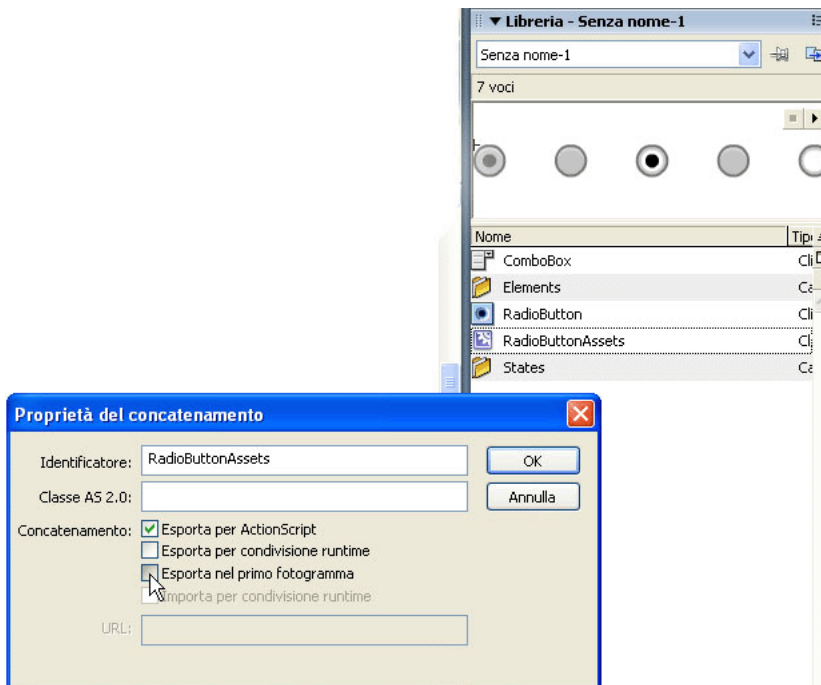
NOTA

L'anteprima dal vivo dei componenti sullo stage non mostra il nuovo tema.

4. Se sono state trascinate nel pannello Libreria le cartelle Assets di singoli componenti, assicurarsi che il simbolo Assets di ogni componente sia impostato su Esporta nel primo fotogramma.

Ad esempio, la cartella Assets del componente RadioButton è denominata RadioButton Assets. Aprire la cartella RadioButton Assets; viene visualizzato il simbolo di un clip filmato denominato RadioButtonAssets. Il simbolo RadioButtonAssets contiene tutti i simboli delle singole risorse.

Fare clic con il pulsante destro del mouse (Windows) o premere Ctrl e fare clic (Macintosh) sul simbolo RadioButtonAssets nella libreria del documento, quindi selezionare l'opzione di menu Concatenamento. Selezionare Esporta nel primo fotogramma in modo da esportare nel primo fotogramma anche tutti i simboli delle singole risorse. Fare quindi clic su OK per salvare le impostazioni.



5. Selezionare Controllo > Prova filmato per vedere il documento dopo l'applicazione del nuovo tema.

Creazione di un nuovo tema

Se non si desidera utilizzare il tema Halo o il tema Sample, è possibile modificarli per crearne uno nuovo.

Alcuni skin contenuti nei temi hanno dimensioni fisse. Aumentando o diminuendo le dimensioni degli skin, i componenti vengono automaticamente ridimensionati per adattarsi. Altri skin sono composti da più parti, alcune statiche alcune che si allungano.

Alcuni skin, ad esempio RectBorder e ButtonSkin, utilizzano l'API di disegno ActionScript per disegnare gli elementi grafici, in quanto è più efficiente sia in termini di dimensioni che di prestazioni. In questi skin è possibile usare come modello il codice ActionScript per adattarli alle proprie esigenze.

Per un elenco di skin supportati da ogni componente e delle relative proprietà, consultare la *Guida di riferimento dei componenti*.

Per creare un nuovo tema:

1. Selezionare il file FLA del tema da utilizzare come modello e farne una copia.
Assegnare alla copia un nome univoco, ad esempio **MyTheme.fla**.
2. Selezionare File > Apri MyTheme.fla in Flash.
3. Selezionare Finestra > Libreria per aprire la libreria nel caso non sia già aperta.
4. Fare doppio clic sul simbolo di uno skin per aprirlo in modalità di modifica del simbolo.
Gli skin si trovano nella cartella Flash UI Components 2/Themes/MMDefault/
Componente Assets (in questo esempio, RadioButton Assets).
5. Modificare il simbolo oppure eliminare la grafica e crearne una nuova.
Per aumentare il livello di ingrandimento, selezionare Visualizza > Ingrandisci. Quando si modifica uno skin, è necessario mantenerne il punto di registrazione per assicurare una visualizzazione corretta. L'angolo superiore sinistro di tutti i simboli modificati deve essere impostato su (0,0).
Ad esempio, aprire la risorsa States/RadioFalseDisabled e assegnare al cerchio interno il colore grigio chiaro.
6. Al termine della modifica del simbolo di skin, fare clic sul pulsante Indietro, sul lato sinistro della barra delle informazioni nella parte superiore dello stage, per tornare alla modalità di modifica del documento.
7. Ripetere le operazioni elencate ai punti da 4 a 6 finché non sono stati modificati tutti gli skin desiderati.
8. Applicare il file MyTheme.fla a un documento seguendo la procedura descritta più avanti in questo capitolo. Vedere [“Applicazione di un nuovo tema a un documento”](#) a pagina 120.

Modifica dei valori predefiniti delle proprietà di stile di un tema

I valori predefiniti delle proprietà di stile vengono forniti da ogni tema in una classe denominata Defaults. Per modificare i valori predefiniti di un tema personalizzato, creare una nuova classe ActionScript denominata Defaults in un pacchetto adeguato al tema e modificare le impostazioni predefinite nel modo desiderato.

Per modificare i valori predefiniti dello stile in un tema:

1. Creare una nuova cartella per il tema in First Run/Classes/mx/skins.
Ad esempio, una cartella denominata **myTheme**.
2. Copiare una classe Defaults esistente nella nuova cartella dei temi.
Ad esempio, copiare mx/skins/halo/Defaults.as in mx/skins/myTheme/Defaults.as.
3. Aprire la nuova classe Defaults nell'Editor di ActionScript.
Gli utenti di Flash Professional 8 possono aprire il file all'interno di Flash, oppure nel Blocco note in Windows o in SimpleText in Macintosh.
4. Modificare la dichiarazione della classe in modo adeguato per il nuovo pacchetto.
Ad esempio, in questo caso, la nuova dichiarazione della classe è `class mx.skins.myTheme.Defaults`.
5. Modificare le impostazioni dello stile nel modo desiderato.
Ad esempio, cambiare il colore predefinito per la disattivazione in rosso scuro.
`o.disabledColor = 0x663333;`
6. Salvare il file di classe Defaults modificato.
7. Copiare una classe FocusRect esistente dal tema di origine al tema personalizzato.
Ad esempio, copiare mx/skins/halo/FocusRect.as in mx/skins/myTheme/FocusRect.as.
8. Aprire la nuova classe FocusRect nell'Editor di ActionScript.
9. Sostituire tutti i riferimenti al pacchetto del tema di origine con i riferimenti al pacchetto del nuovo tema.
Ad esempio, sostituire tutte le occorrenze di "halo" con "myTheme".
10. Salvare il file di classe FocusRect modificato.
11. Aprire il file FLA del tema personalizzato.
Questo esempio utilizza MyTheme fla.
12. Aprire la libreria (Finestra > Libreria) e individuare il simbolo Defaults.
In questo esempio, si trova in Flash UI Components 2/Themes/MMDefault/Defaults.
13. Modificare le proprietà del simbolo Defaults.

14. Modificare l'impostazione Classe AS 2.0 in modo adeguato per il nuovo pacchetto.

La classe di esempio è `mx.skins.myTheme.Defaults`.

15. Fare clic su OK.

16. Individuare il simbolo FocusRect.

In questo esempio, si trova in Flash UI Components 2/Themes/MMDefault/FocusRect.

17. Modificare le proprietà del simbolo FocusRect.

18. Modificare l'impostazione Classe AS 2.0 in modo adeguato per il nuovo pacchetto.

La classe di esempio è `mx.skins.myTheme.FocusRect`.

19. Fare clic su OK.

20. Applicare il tema personalizzato a un documento seguendo la procedura descritta nella sezione seguente

Quando si trascinano nel documento di destinazione le risorse dal tema predefinito, includere i simboli Defaults e FocusRect.

In questo esempio è stato utilizzato un nuovo tema per personalizzare il colore del testo dei componenti disattivati. Questo tipo specifico di personalizzazione, ovvero la modifica di un singolo valore predefinito di una proprietà di stile, sarebbe stato più semplice applicando gli stili come illustrato in [“Uso degli stili per personalizzare il testo e il colore dei componenti” a pagina 86](#). L'uso di un nuovo tema per personalizzare le impostazioni predefinite è appropriato quando si desidera personalizzare molte proprietà di stile o quando la creazione di un nuovo tema è necessaria per personalizzare gli elementi grafici del componente.

Applicazione di un nuovo tema a un documento

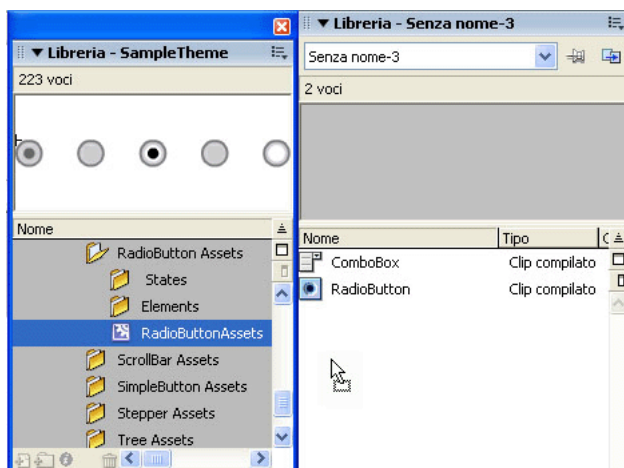
Per applicare un nuovo tema a un documento, aprire il file FLA di un tema come libreria esterna e trascinare nella libreria del documento la cartella dei temi dalla libreria esterna. Nella procedura seguente viene spiegato il processo in dettaglio, presupponendo che sia già disponibile un nuovo tema. Per ulteriori informazioni, vedere [“Creazione di un nuovo tema” a pagina 118](#).

Per applicare un tema a un documento:

1. Selezionare File > Apri, quindi aprire il documento che utilizza i componenti della versione 2 in Flash oppure selezionare File > Nuovo e creare un documento nuovo che utilizzi i componenti della versione 2.
2. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file FLA del tema da applicare al documento.

3. Nel pannello Libreria del tema, selezionare Flash UI Components 2/Themes/MMDefault, quindi trascinare nella libreria del documento le cartelle Assets di qualsiasi componente si desideri utilizzare.

Ad esempio, trascinare nella libreria la cartella RadioButton Assets.



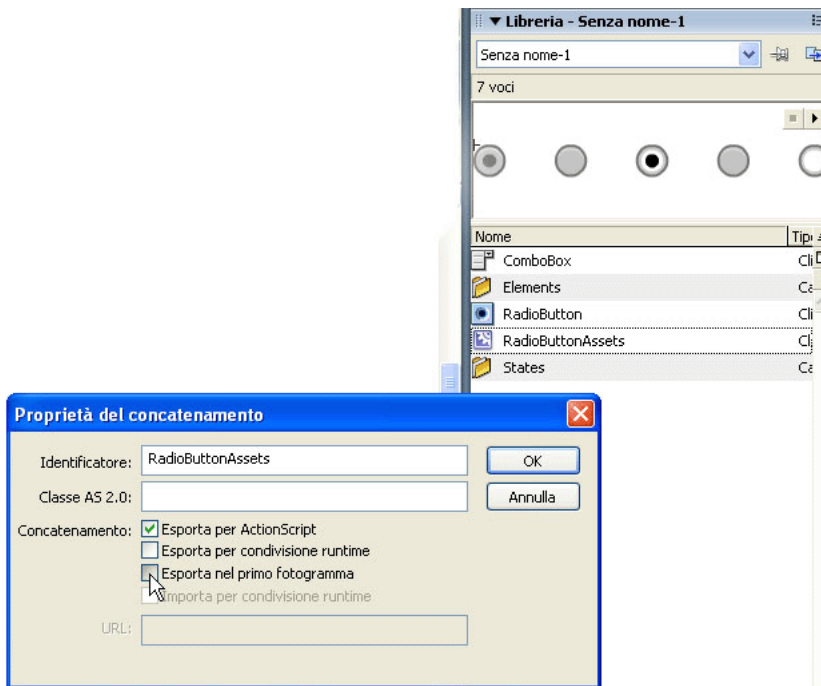
Se non si è certi dei componenti presenti nel documento, trascinare l'intero clip filmato del tema (ad esempio, per il file SampleTheme.fla, il clip filmato del tema principale è Flash UI Components 2 > SampleTheme) nello stage. Gli skin vengono assegnati automaticamente ai componenti del documento.

NOTA

L'anteprima dal vivo dei componenti sullo stage non mostra il nuovo tema.

4. Se sono state trascinate nella libreria ThemeApply.fla le cartelle Assets dei singoli componenti, assicurarsi che il simbolo Assets di ogni componente sia impostato su Esporta nel primo fotogramma.

Ad esempio, la cartella Assets del componente RadioButton è denominata RadioButtonAssets e contiene un simbolo denominato RadioButtonAssets che a sua volta contiene tutti i simboli delle singole risorse. Se si imposta Esporta nel primo fotogramma per il simbolo RadioButtonAssets, vengono esportati nel primo fotogramma anche tutti i simboli delle singole risorse.



5. Selezionare Controllo > Prova filmato per vedere applicato il nuovo tema.

Modifica delle impostazioni di esportazione

Quando si applica il tema Sample o Halo al documento, molte delle risorse degli skin sono impostate per l'esportazione nel primo fotogramma allo scopo di renderle immediatamente disponibili ai componenti durante la riproduzione. Tuttavia, se si modificano le impostazioni di esportazione di pubblicazione (File > Impostazioni di pubblicazione > scheda Flash > pulsante Impostazioni della Versione di ActionScript > Esporta fotogramma per le classi) del file FLA impostandole su un fotogramma successivo al primo, è inoltre necessario modificare le impostazioni di esportazione per le risorse nei temi Sample e e Halo. A questo scopo, è necessario aprire le risorse dei componenti seguenti nella libreria del documento e deselezionare la casella di controllo Esporta nel primo fotogramma (fare clic con il pulsante destro del mouse su > Concatenamento > Esporta nel primo fotogramma):

Sample, tema

- 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

Halo, tema

- 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

Combinazione di skin e di stili per personalizzare un componente

In questa sezione si procederà a personalizzare un'istanza del componente combobox utilizzando impostazioni relative a stili, temi e skin. Le procedure illustrano come combinare le impostazioni relative a skin e stili per creare una presentazione originale per un componente.

Creazione di un'istanza del componente nello stage

La prima parte di questo esercizio richiede la creazione di un'istanza ComboBox da personalizzare.

Per creare l'istanza ComboBox:

1. Trascinare un componente ComboBox nello stage.
2. Nella finestra di ispezione Proprietà, denominare l'istanza **my_cb**.
3. Nel primo fotogramma della linea temporale principale, aggiungere il codice ActionScript seguente (assicurandosi di aggiungerlo al fotogramma e non al componente; la barra del titolo del pannello Azioni deve indicare "Azioni - Fotogramma"):

```
my_cb.addItem({data:1, label:"One"});
my_cb.addItem({data:2, label:"Two"});
```
4. Selezionare Controllo > Prova filmato per visualizzare la casella combinata con lo stile e lo skin predefiniti del tema Halo.

Creazione della nuova dichiarazione di stile

È ora necessario creare una nuova dichiarazione di stile e assegnare gli stili alla dichiarazione. Dopo aver inserito tutti gli stili desiderati nella dichiarazione di stile, è possibile assegnare il nuovo nome di stile all'istanza ComboBox.

Per creare una nuova dichiarazione di stile e assegnarle un nome:

1. Nel primo fotogramma della linea temporale principale, aggiungere la riga seguente all'inizio del codice ActionScript (come convenzione di codifica, è consigliabile inserire tutte le istruzioni di importazione all'inizio del codice ActionScript):

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

2. Sulla riga successiva, denominare la nuova dichiarazione di stile e aggiungerla alle definizioni di stile globali:

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

Dopo aver assegnato una nuova dichiarazione di stile al foglio di stile `_global`, è possibile associare singole impostazioni di stile alla dichiarazione di stile `new_style`. Per ulteriori informazioni sulla creazione di un foglio di stile per gruppi di componenti, anziché di definizioni di stile per una singola istanza, vedere [“Impostazione di stili personalizzati per gruppi di componenti” a pagina 92](#)).

3. Associare alcune impostazioni di stile alla dichiarazione di stile `new_style`. Le impostazioni di stile seguenti includono le definizioni di stile disponibili per il componente ComboBox (vedere “Uso degli stili con il componente ComboBox” nella *Guida di riferimento dei componenti* per un elenco più completo) nonché gli stili della classe `RectBorder`, dato che il componente ComboBox utilizza tale classe:

```
new_style.setStyle("textAlign", "right");  
new_style.setStyle("selectionColor", "white");  
new_style.setStyle("useRollOver", false);  
// borderStyle from RectBorder class  
new_style.setStyle("borderStyle", "none");
```

Assegnazione di definizioni di stile alla casella combinata

A questo punto, è disponibile una dichiarazione di stile che include vari stili, ma è necessario assegnare in modo esplicito il nome dello stile all'istanza del componente. È possibile assegnare questa nuova dichiarazione di stile a *qualsiasi* istanza del componente all'interno del documento nel modo indicato di seguito. Aggiungere la riga seguente dopo le istruzioni addItem() per my_cb (come convenzione di codifica, è consigliabile mantenere raggruppate tutte le istruzioni di costruzione della casella combinata):

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

Il codice ActionScript associato al primo fotogramma della linea temporale principale dovrebbe essere analogo al seguente:

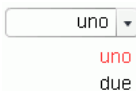
```
import mx.styles.CSSStyleDeclaration;

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 from RectBorder class
new_style.setStyle("borderStyle", "none");

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

Selezionare Controllo > Prova filmato per visualizzare la casella combinata con gli stili applicati:



Modifica del tema della casella combinata

Ogni componente dell'interfaccia utente elenca le proprietà di stile che è possibile impostare per il componente stesso (ad esempio, tutte le proprietà di stile che è possibile impostare per un componente ComboBox sono indicate in “Personalizzazione del componente ComboBox” nella *Guida di riferimento dei componenti*). All'interno della tabella delle proprietà dello stile, una colonna denominata "Tema" mostra quale tema installato supporta ciascuna proprietà dello stile. Non tutte le proprietà dello stile sono supportate da tutti i temi installati. Il tema predefinito per tutti i componenti dell'interfaccia utente è il tema Halo. Quando si modifica il tema impostandolo su Sample, è possibile utilizzare un diverso set di proprietà dello stile (alcune proprietà potrebbero non essere più disponibili se elencate solo come supportate da Halo).

Per modificare il tema per il componente con lo stile applicato:

1. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file SampleTheme.fla per aprire la libreria del tema Sample in Flash.

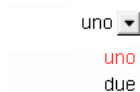
Questo file si trova nella cartella Configuration dell'applicazione:

- In Windows: In Windows: C:\Programmi\Macromedia\Flash 8\lingua\Configuration\ComponentsFLA\
- In Macintosh: HD/Applicazioni/Macromedia Flash 8/Configuration/ComponentFLA/

2. Trascinare il clip filmato SampleTheme (Flash UI Components 2 > SampleTheme) principale dalla libreria SampleTheme nella libreria del documento.

Il componente ComboBox è una combinazione di varie classi e componenti e richiede risorse da questi altri componenti, incluse Border e ScrollBar. Il modo più semplice per assicurarsi di disporre di tutte le risorse necessarie per un tema è trascinare tutte le risorse del tema nella libreria.

3. Selezionare Controllo > Prova filmato per visualizzare la casella combinata con gli stili applicati:



Modifica delle risorse dello skin della casella combinata

Per modificare l'aspetto di un componente, modificare gli skin che includono il componente, in modalità grafica. Per modificare gli skin, aprire le risorse grafiche del componente dall'interno del tema corrente e modificare i simboli per il componente. Macromedia consiglia questo approccio dato che non determina la rimozione o l'aggiunta di simboli che potrebbero essere necessari per altri componenti; questo approccio modifica l'aspetto di un simbolo di skin di componente esistente.

NOTA

È possibile, ma sconsigliato, modificare i file della classe di origine per un componente in modo che utilizzi i simboli con nomi diversi come skin, ed è possibile modificare a livello di codice il codice ActionScript all'interno di un simbolo di skin (per un esempio di simboli di skin ActionScript personalizzati, vedere “Personalizzazione del componente Accordion (solo Flash Professional)” nella *Guida di riferimento dei componenti*). Tuttavia, dato che vari componenti, incluso il componente ComboBox, condividono risorse con altri, la modifica dei file sorgente o dei nomi dei simboli di skin potrebbe generare risultati imprevisti.

Quando si modifica un simbolo di skin del componente:

- Tutte le istanze del componente utilizzeranno i nuovi skin (ma non gli stili personalizzati salvo se associati esplicitamente alle istanze), e alcuni componenti dipendenti dal componente utilizzeranno i nuovi skin.
- Se si assegna un nuovo tema dopo aver modificato gli skin del componente, assicurarsi di non sovrascrivere gli skin "modificati" esistenti (una finestra di dialogo richiede se si desidera sovrascrivere gli skin e consente di impedire la sovrascrittura).

In questa sezione, si continuerà a utilizzare la casella combinata della sezione precedente (vedere “[Modifica del tema della casella combinata](#)” a pagina 128). Nei passaggi seguenti l'aspetto della freccia rivolta verso il basso che consente di aprire il menu della casella combinata viene modificato da freccia a cerchio.

Per modificare il simbolo Freccia giù della casella combinata:

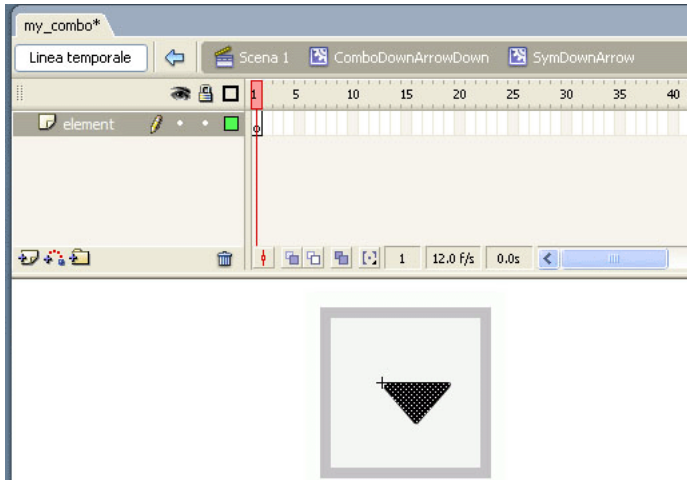
1. Nella libreria del documento, aprire le risorse di ComboBox per vedere i clip filmato che rappresentano gli skin per il pulsante che apre e chiude l'istanza della casella combinata in fase di runtime. In particolare, aprire la cartella Themes > MMDefault > ComboBox Assets > States nella libreria del documento.

La cartella States include quattro clip filmato: ComboDownArrowDisabled, ComboDownArrowDown, ComboDownArrowOver e ComboDownArrowUp. Tutti e quattro questi simboli sono composti da altri simboli e tutti e quattro utilizzano lo stesso simbolo per la Freccia giù (triangolo), denominato SymDownArrow.

2. Fare doppio clic sul simbolo ComboDownArrowDown per modificarlo.
Potrebbe essere necessario ingrandire, fino all'800%, per vedere i dettagli del pulsante.
3. Fare doppio clic sulla freccia giù (triangolo bianco) per modificarla.

NOTA

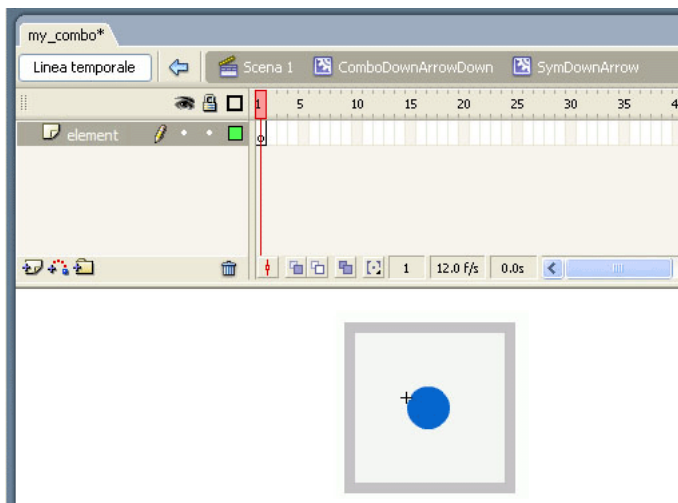
Assicurarsi che il simbolo SymDownArrow sia selezionato, in modo da eliminare solo la forma all'interno del clip filmato e non il simbolo del clip filmato.



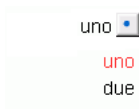
4. Eliminare la freccia rivolta verso il basso selezionata (il triangolo nero, *non l'intero clip filmato*) nello stage.

5. Mentre si modifica SymDownArrow, disegnare un cerchio al posto della freccia rivolta verso il basso.

Per rendere la modifica più evidente, è possibile disegnare un cerchio di un colore vivace, ad esempio blu, di circa 4 pixel x 4 pixel, con coordinata x 0 e coordinata y -1, in modo che sia centrato.



6. Selezionare Controllo > Prova filmato per visualizzare la casella combinata con lo skin applicato:



Nella libreria del documento, se si seleziona ComboDownArrowOver e ComboDownArrowUp, si noterà che anch'essi includono il cerchio blu anziché il triangolo bianco, dato che anch'essi utilizzano SymDownArrow come simbolo per la freccia rivolta verso il basso.

In questo capitolo vengono illustrate le procedure per creare un componente personalizzato e per inserirlo in un pacchetto per la distribuzione.

Questo capitolo contiene le seguenti sezioni:

File di origine dei componenti	134
Panoramica sulla struttura dei componenti	134
Creazione del primo componente	135
Selezione di una classe principale	145
Creazione del clip filmato di un componente	149
Creazione del file di classe ActionScript	154
Come incorporare componenti esistenti in un componente	185
Esportazione e distribuzione di un componente	194
Passaggi finali dello sviluppo di un componente	198

File di origine dei componenti

I componenti disponibili nel pannello Componenti sono clip SWC precompilati. Sono inoltre inclusi il documento Flash (FLA) di origine che contiene la grafica e i file di classe ActionScript (AS) di origine contenenti il codice di questi componenti, in modo che sia possibile utilizzarli nella creazione di componenti personalizzati. I file di origine dei componenti della versione 2 vengono installati con Macromedia Flash. Prima di creare componenti personalizzati, è utile aprire alcuni di questi file per osservarne il contenuto e cercare di comprenderne la struttura. Il componente RadioButton è un ottimo esempio di componente semplice, che si consiglia di esaminare per primo. Tutti i componenti sono rappresentati da simboli nella libreria StandardComponents.fla. Ogni simbolo è collegato a una classe ActionScript. Il percorso è il seguente:

- Codice di origine dei file FLA
 - In Windows: C:\Programmi\Macromedia\Flex 2\lingua\Configuration\ComponentFLA\StandardComponents.fla.
 - In Macintosh: HD/Applicazioni/Macromedia Flash 8/Configuration/ComponentFLA/StandardComponents.fla
- File delle classi ActionScript
 - In Windows: C:\Programmi\Macromedia\Flex 2\lingua\First Run\Classes\mx
 - In Macintosh: HD/Applicazioni/Macromedia Flash 8/First Run/Classes/mx

Panoramica sulla struttura dei componenti

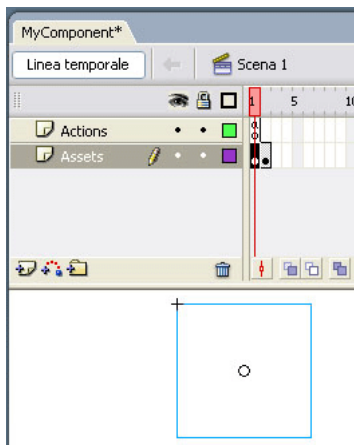
Un componente è formato da un file Flash (FLA) e da un file ActionScript (AS). Altri file, ad esempio un'icona o un file di debug .swd, possono essere creati facoltativamente e inclusi nel pacchetto insieme al componente, ma tutti i componenti richiedono in ogni caso un file FLA e un file AS. Dopo aver sviluppato un componente, è necessario esportarlo in un file SWC.



Un file Flash (FLA), un file ActionScript (AS) e un file SWC

Il file FLA contiene un simbolo di clip filmato che deve essere collegato al file AS nelle finestre di dialogo Proprietà del concatenamento e Definizione del componente.

Il simbolo del clip filmato dispone di due fotogrammi e due livelli. Al primo livello, Actions, è associata una funzione globale `stop()` sul fotogramma 1. Il secondo livello, Assets, è dotato di due fotogrammi chiave: il fotogramma 1 contiene un riquadro di delimitazione. Il fotogramma 2 contiene tutte le altre risorse utilizzate dal componente, inclusi gli elementi grafici e le classi base.



Il codice ActionScript che specifica le proprietà e i metodi del componente si trova in un file di classe ActionScript separato. Il file di classe definisce inoltre quali classi sono eventualmente estese dal componente. Il nome del file di classe AS corrisponde al nome del componente seguito dall'estensione ".as". Ad esempio, `MyComponent.as` contiene il codice di origine del componente `MyComponent`.

È consigliabile salvare i file FLA e AS del componente nella stessa cartella, utilizzando per entrambi lo stesso nome. Se il file AS non viene salvato nella stessa cartella, verificare che la cartella si trovi nel percorso di classe per consentirne l'individuazione da parte del file FLA. Per ulteriori informazioni sul percorso di classe, vedere "Classi" in *Apprendimento di ActionScript 2.0 in Flash*.

Creazione del primo componente

In questa sezione verrà descritto come creare un componente Dial (Quadrante). I file del componente completi, `Dial.fla`, `Dial.as`, `Dial.swf` e `DialAssets.fla`, si trovano nella cartella degli esempi sul computer:

- In Windows: cartella `C:\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\DialComponent`.
- In Macintosh: cartella `HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/DialComponent`.

Il componente Dial è un potenziometro, simile a quelli utilizzati per misurare la differenza di potenziale della tensione elettrica. È possibile fare clic sulla lancetta e trascinarla per modificarne la posizione. L'API del componente Dial dispone di una proprietà, `value`, che può essere utilizzata per impostare la posizione della lancetta mediante `Get` e `Set`.

Questa sezione illustra la procedura di creazione di un componente. Queste procedure sono illustrate in maggiore dettaglio nelle sezioni seguenti (tra le quali [“Selezione di una classe principale”](#) a pagina 145, [“Creazione del clip filmato di un componente”](#) a pagina 149, [“Creazione del file di classe `ActionScript`”](#) a pagina 154 e [“Esportazione e distribuzione di un componente”](#) a pagina 194). La sezione contiene i seguenti argomenti:

- [“Creazione del file Flash \(FLA\) Dial”](#) a pagina 136
- [“Creazione del file di classe Dial”](#) a pagina 139
- [“Prova ed esportazione del componente Dial”](#) a pagina 143

Creazione del file Flash (FLA) Dial

I primi passaggi per la creazione di un componente includono la creazione del clip filmato del componente all'interno di un file di documento FLA.

Per creare il file FLA Dial:

1. In Flash, selezionare **File > Nuovo** e creare un nuovo documento.
2. Scegliere **> Salva con nome** e salvare il file come **Dial.fla**.
È possibile utilizzare un nome qualsiasi, tuttavia è più pratico assegnare al file lo stesso nome del componente.
3. Selezionare **Inserisci > Nuovo simbolo**. Il componente stesso viene creato come un nuovo simbolo `MovieClip`, in modo che sia disponibile nella libreria.
Denominare il componente **Dial** e assegnarlo al comportamento **Clip filmato**.
4. Se la sezione **Concatenamento** della finestra di dialogo **Crea un nuovo simbolo** non è visibile, fare clic sul pulsante **Avanzato** per aprirla.
5. Nell'area **Concatenamento** selezionare **Esporta per `ActionScript`** e **deselezionare `Esporta nel primo fotogramma`**.
6. Nella casella di testo **Identificatore**, immettere un identificatore di concatenamento come **Dial_ID**.
7. Nella casella di testo **Classe AS 2.0**, immettere **Dial**. Questo valore è il nome di classe del componente. Se la classe è inclusa in un pacchetto, ad esempio `mx.controls.Button`, specificare il nome completo del pacchetto.
8. Fare clic su **OK**.
Flash passa alla modalità di modifica dei simboli.

9. Inserire un nuovo livello. Assegnare al livello superiore il nome **Actions** e al livello inferiore il nome **Assets**.

10. Selezionare il fotogramma 2 nel livello Assets e inserire un fotogramma chiave (F6).

Questa è la struttura del clip filmato del componente: un livello Actions e un livello Assets. Nel livello Actions è presente un fotogramma chiave; nel livello Assets sono presenti 2 fotogrammi chiave.

11. Selezionare il fotogramma 1 nel livello Actions e aprire il pannello Azioni (F9). Immettere una funzione globale `stop();`.

In questo modo si impedisce al clip filmato di passare al fotogramma 2.

12. Scegliere File > Importa > Apri libreria esterna e selezionare il file StandardComponents.fla dalla cartella Configuration/ComponentFLA. Ad esempio:

- In Windows: C:\Programmi\Macromedia\Flash 8\lingua\Configuration\ComponentFLA\StandardComponents.fla.
- In Macintosh: HD/Applicazioni/Macromedia Flash 8/Configuration/ComponentFLA/StandardComponents.fla

NOTA

Per ulteriori informazioni sulle posizioni delle cartelle, vedere "Cartelle di configurazione installate con Flash" in *Guida introduttiva di Flash*.

13. Il componente Dial estende la classe base UIComponent, pertanto è necessario trascinare un'istanza di UIComponent nel documento Dial. Nella libreria StandardComponents.fla, passare al clip filmato UIComponent nella seguente cartella: Flash UI Components 2 > Base Classes > FUIObject Subclasses e trascinare la cartella nella libreria Dial.fla.

Le dipendenze di risorse vengono copiate nella libreria di Dial con UIComponent.

NOTA

Quando si trascina UIComponent nella libreria di Dial, viene modificata la gerarchia delle cartelle all'interno della libreria. Se si prevede di riutilizzare la libreria oppure di utilizzarla con altri gruppi di componenti, ad esempio i componenti della versione 2, ristrutturare la gerarchia delle cartelle affinché corrisponda alla libreria StandardComponents.fla. In questo modo, la libreria viene organizzata nel modo corretto e si evitano simboli duplicati.

14. Nel livello Assets, selezionare il fotogramma 2, quindi trascinare un'istanza di UIComponent allo stage.

15. Chiudere la libreria StandardComponents.fla.

16. Selezionare File > Importa > Apri libreria esterna, quindi selezionare il file DialAssets.fla.
- In Windows: C:\Programmi\Macromedia\Flex 8\Samples and Tutorials\Samples\Components\DialComponent\DialAssets.fla.
 - In Macintosh: HD/Applicazioni/Macromedia Flex 8/Samples and Tutorials/Samples/Components/DialComponent/DialAssets.fla
17. Nel livello Assets, selezionare il fotogramma 2, quindi trascinare un'istanza del clip filmato DialFinal dalla libreria DialAssets allo stage.

Tutte le risorse del componente vengono aggiunte al fotogramma 2 del livello Assets. Poiché sul fotogramma 1 del livello Actions è presente una funzione globale `stop()`, le risorse nel fotogramma 2 non sono visibili nel modo in cui sono disposte sullo stage.

Le risorse vengono aggiunte al fotogramma 2 per due motivi:

- Per copiare automaticamente nella libreria tutte le risorse e le risorse secondarie e renderle disponibili per crearne un'istanza dinamicamente (nel caso di DialFinal) oppure per accedere ai metodi, alle proprietà e agli eventi delle risorse stesse (nel caso di UIComponent).
- L'inserimento delle risorse in un fotogramma fa sì che le risorse vengano caricate in maniera più fluida durante lo streaming del filmato, evitando di impostare l'esportazione delle risorse nella libreria prima del primo fotogramma. Questo metodo impedisce che si verifichi un picco di elaborazione iniziale durante lo scaricamento.

18. Chiudere la libreria DialAssets.fla.

19. Nel livello Assets, selezionare il fotogramma 1. Trascinare il clip filmato BoundingBox dalla libreria (cartella Flex UI Components 2 > Component Assets) sullo stage. Assegnare all'istanza BoundingBox il nome **boundingBox_mc**. Utilizzare il pannello Informazioni per impostare altezza e larghezza del clip filmato DialFinal su 250 pixel, e le coordinate x,y su 0, 0.

L'istanza BoundingBox viene utilizzata per creare l'anteprima dal vivo del componente e per ridimensionarlo durante la creazione. Le dimensioni del riquadro di delimitazione devono essere tali da poter racchiudere tutti gli elementi grafici presenti nel componente.

NOTA

Se si estende un componente, incluso qualsiasi componente della versione 2, è necessario mantenere gli eventuali nomi di istanze già utilizzati da quel componente, in quanto il codice fa riferimento ad essi. Se, ad esempio, si include un componente della versione 2 che utilizza già il nome di istanza `boundingBox_mc`, evitare di modificare il nome. Per i nomi di istanza personalizzati, è possibile utilizzare qualsiasi nome univoco che non crei conflitti con un nome già esistente nella stessa area di validità.

20. Selezionare il clip filmato Dial nella libreria e scegliere Definizione del componente dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse; Mac: clic tenendo premuto il tasto Control).
21. Nella casella di testo Classe AS 2.0, immettere **Dial**.

Questo valore è il nome di classe ActionScript. Se la classe è inclusa in un pacchetto, il valore è il nome completo del pacchetto, ad esempio mx.controls.CheckBox.
22. Fare clic su OK.
23. Salvare il file.

Creazione del file di classe Dial

È ora necessario creare il file di classe Dial come un nuovo file ActionScript.

Per creare il file di classe Dial:

1. In Flash, selezionare File > Nuovo, quindi selezionare File ActionScript.
2. Selezionare File > Salva con nome e salvare il file con il nome **Dial.as** nella stessa cartella che contiene il file Dial fla.

NOTA

È possibile utilizzare qualsiasi editor di testo per salvare il file Dial.as.

3. È possibile copiare o digitare nel nuovo file Dial.as il seguente codice di classe ActionScript del componente Dial. Se si digita il codice, invece di copiarlo, è più facile acquisire velocemente dimestichezza con ogni elemento del codice del componente.

Per una descrizione di ogni sezione, leggere i commenti all'interno del codice stesso. Per informazioni dettagliate sugli elementi di un file di classe di un componente, vedere [“Panoramica del file di classe di un componente” a pagina 155](#).

```
// Importa il pacchetto in modo che sia possibile fare riferimento
// direttamente alla classe.
import mx.core.UIComponent;

// Tag per metadati Event
[Event("change")]
class Dial extends UIComponent
{
    // I componenti devono dichiarare questi valori per essere
    // correttamente
    // rappresentati nell'architettura dei componenti.
    static var symbolName:String = "Dial";
    static var symbolOwner:Object = Dial;
    var className:String = "Dial";
```

```

// I clip filmato del quadrante e della lancetta corrispondenti
// alla rappresentazione grafica del componente
private var needle:MovieClip;
private var dial:MovieClip;
private var boundingBox_mc:MovieClip;

// La variabile del membro privato "__value" è pubblicamente
// accessibile mediante i metodi getter/setter impliciti,
// L'aggiornamento di questa proprietà aggiorna la posizione della
// lancetta
// quando viene impostato il valore.
private var __value:Number = 0;

// Questo flag viene impostato quando l'utente trascina la
// lancetta con il mouse e viene cancellato successivamente.
private var dragging:Boolean = false;

// Funzione di costruzione;
// Sebbene richiesti per tutte le classi, i componenti della versione
// 2 richiedono
// che la funzione di costruzione sia vuota, con zero argomenti.
// Tutte le operazioni di inizializzazione vengono eseguite in un
// metodo init()
// obbligatorio dopo la costruzione dell'istanza della classe.
function Dial() {
}

// Codice di inizializzazione:
// Il metodo init() è obbligatorio per i componenti della versione 2.
// Deve inoltre chiamare
// a sua volta il metodo init() della classe principale con
// super.init().
// Il metodo init() è obbligatorio per i componenti che estendono
// UIComponent.
function init():Void {
    super.init();
    useHandCursor = false;
    boundingBox_mc._visible = false;
    boundingBox_mc._width = 0;
    boundingBox_mc._height = 0;
}

// Crea gli oggetti secondari richiesti all'avvio:
// Il metodo createChildren() è obbligatorio per i componenti
// che estendono UIComponent.
public function createChildren():Void {
    dial = createObject("DialFinal", "dial", 10);
    size();
}

```

```

// Il metodo draw() è obbligatorio per i componenti della versione 2.
// Viene chiamato dopo che il componente è
// stato invalidato mediante una chiamata a invalidate().
// È preferibile rispetto al ridisegno dall'interno della funzione
set()
// per il valore; se sono presenti altre proprietà, è infatti
// opportuno includere le modifiche in un'operazione batch di
ridisegno,
// anziché effettuarle individualmente. Questo approccio consente
// una maggiore efficienza e una migliore centralizzazione del codice.
function draw():Void {
    super.draw();
    dial.needle._rotation = value;
}

// Il metodo size() viene richiamato quando le dimensioni del
componente
// cambiano. A questo punto, è possibile ridimensionare gli elementi
secondari
// e la grafica del quadrante e della lancetta.
// Il metodo size() è obbligatorio per i componenti che estendono
UIComponent.
function size():Void {
    super.size();
    dial._width = width;
    dial._height = height;
    // Determina, se necessario, il ridisegno della lancetta.
    invalidate();
}

// Questa è la funzione getter/setter della proprietà value.
// I metadati [Inspectable] consentono la visualizzazione della
proprietà
// nella finestra di ispezione Proprietà. Questa è una funzione
getter/setter
// che consente di chiamare invalidate e forzare il
// ridisegno da parte del componente quando viene modificato il
valore.
[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 di attendere i clic del mouse
function onPress()
{
    beginDrag();
}

// Quando si fa clic sul quadrante, viene impostato il flag di
// trascinamento.
// Vengono assegnate funzioni di callback agli eventi del mouse.
function beginDrag()
{
    dragging = true;
    onMouseMove = mouseMoveHandler;
    onMouseUp = mouseUpHandler;
}

// Rimuove gli eventi del mouse al termine del trascinamento
// e cancella il flag
function mouseUpHandler()
{
    dragging = false;
    delete onMouseMove;
    delete onMouseUp;
}

function mouseMoveHandler()
{
    // Calcola l'angolo
    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"} );
        }
    }
}
}

```

Prova ed esportazione del componente Dial

È stato creato il file Flash contenente gli elementi grafici, le classi base e il file di classe che comprendono tutte le funzionalità del componente Dial. A questo punto, è possibile provare il componente.

Idealmente, la prova del componente dovrebbe essere effettuata durante la creazione, specialmente durante la scrittura del file di classe. Il modo più veloce per effettuare una prova consiste nel convertire il componente in un clip compilato e utilizzarlo nel file FLA del componente.

Dopo aver completato lo sviluppo di un componente, esportarlo in un file SWC. Per ulteriori informazioni, vedere [“Esportazione e distribuzione di un componente” a pagina 194](#).

Per provare il componente Dial:

1. Nel file Dial.fla selezionare il componente Dial nella libreria, aprire il menu di scelta rapida Libreria (Windows: clic con il pulsante destro del mouse; Mac: clic tenendo premuto il tasto Control), e selezionare Conversione in clip compilato.

Alla libreria viene aggiunto un clip compilato, denominato Dial SWF.

NOTA

Se è già stato creato un clip compilato, ad esempio se è la seconda o la terza volta che si esegue la prova, viene visualizzata la finestra di dialogo Risolvi il conflitto tra librerie. Scegliere Sostituisci gli elementi esistenti per aggiungere la nuova versione al documento.

2. Trascinare Dial SWF sullo stage nella linea temporale principale.
3. È possibile ridimensionare il clip compilato e impostare la relativa proprietà value nella finestra di ispezione Proprietà o nella finestra di ispezione dei componenti. Quando si imposta la proprietà value, la posizione della lancetta dovrebbe cambiare conseguentemente.

4. Per provare la proprietà `value` in fase di runtime, assegnare il nome di istanza **dial** al quadrante e aggiungere il codice seguente sul fotogramma 1 nella linea temporale principale:

```
// posizione del campo di testo
var textXPos:Number = dial.width/2 + dial.x
var textYPos:Number = dial.height/2 + dial.y;

// crea un campo di testo per la visualizzazione del valore di dial
createTextField("dialValue", 10, textXPos, textYPos, 100, 20);

// crea un listener per gestire l'evento change
function change(evt){
// inserisce la proprietà value nel campo di testo
// ogni volta che la lancetta si sposta
    dialValue.text = dial.value;
}
dial.addEventListener("change", this);
```

5. Scegliere Controllo > Prova filmato per provare il componente in Flash Player.

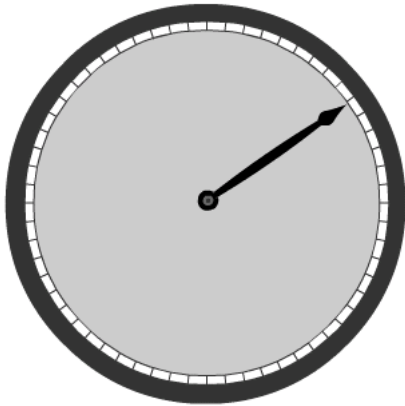
Per esportare il componente Dial:

1. Nel file `Dial.fla` selezionare il componente `Dial` nella libreria, aprire il menu di scelta rapida Libreria (Windows: clic con il pulsante destro del mouse; Mac: clic tenendo premuto il tasto Control), e selezionare Esporta file SWC.
2. Scegliere una posizione in cui salvare il file SWC.

Se il file viene salvato nella cartella Configuration/Components a livello di utente, è possibile ricaricare il pannello Componenti senza riavviare Flash e il componente viene visualizzato nel pannello.

NOTA

Per ulteriori informazioni sulle posizioni delle cartelle, vedere “Cartelle di configurazione installate con Flash” in *Uso di Flash*.



Il componente Dial completo

Selezione di una classe principale

Il primo aspetto da definire quando si crea un componente riguarda la possibilità di estendere una delle classi della versione 2. Se si sceglie questa opzione, è possibile estendere una classe di componenti, ad esempio, Button, CheckBox, ComboBox, List e così via, oppure una delle classi base UIObject o UIComponent. Tutte le classi dei componenti, a eccezione dei componenti Media, estendono le classi base. Se si estende una classe di componenti, si eredita automaticamente anche dalle classi base.

Le due classi base forniscono funzioni comuni per i componenti. Se si estendono queste classi, il componente dispone inizialmente di un insieme base di metodi, proprietà ed eventi.

Nell'architettura della versione 2, non è necessario creare una sottoclasse di `UIObject` o `UIComponent`, o di qualsiasi altra classe. Anche se le classi di componenti ereditano direttamente dalla classe `MovieClip`, è possibile utilizzare molte funzioni dei componenti: esportare un file SWC o un clip compilato, utilizzare l'anteprima dal vivo incorporata, visualizzare le proprietà modificabili e altro. Se, tuttavia, si desidera che i componenti interagiscano con i componenti di Macromedia versione 2, e che vengano utilizzate le classi di gestori, è necessario estendere `UIObject` o `UIComponent`.

Nella tabella seguente è riportata una breve descrizione delle classi base della versione 2:

Classe base	Estende	Descrizione
<code>mx.core.UIObject</code>	<code>MovieClip</code>	<p><code>UIObject</code> è la classe di base di tutti gli oggetti grafici. Può avere una forma, disegnarsi automaticamente ed essere invisibile.</p> <p><code>UIObject</code> fornisce le seguenti funzionalità:</p> <ul style="list-style-type: none"> • Modifica degli stili • Gestione degli eventi • Ridimensionamento tramite modifica in scala
<code>mx.core.UIComponent</code>	<code>UIObject</code>	<p><code>UIComponent</code> è la classe di base di tutti i componenti. <code>UIComponent</code> fornisce le seguenti funzionalità:</p> <ul style="list-style-type: none"> • Navigazione con attivazione del componente • Creazione di uno schema di tabulazione • Abilitazione e disabilitazione dei componenti • Ridimensionamento dei componenti • Gestione degli eventi di livello basso del mouse e della tastiera

Nozioni fondamentali sulla classe `UIObject`

I componenti basati sulla versione 2 di Macromedia Component Architecture derivano dalla classe `UIObject`, che è una sottoclasse della classe `MovieClip`. La classe `MovieClip` è la classe base per tutte le classi di Flash che rappresentano oggetti visivi nelle schermate.

La classe `UIObject` consente di aggiungere metodi per la gestione di stili ed eventi. Invia gli eventi ai propri listener subito prima della realizzazione di un disegno (l'evento `draw` è l'equivalente dell'evento `MovieClip.onEnterFrame`), quando viene caricata e scaricata (`load` e `unload`), quando viene modificato il layout (`move`, `resize`) e quando viene nascosta o visualizzata (`hide` e `reveal`).

`UIObject` fornisce variabili alternative di sola lettura per stabilire la posizione e le dimensioni di un componente (`width`, `height`, `x`, `y`), e i metodi `move()` e `setSize()` per modificare la posizione e le dimensioni di un oggetto.

La [Classe UIObject](#) implementa quanto segue:

- Stili
- Eventi
- Ridimensionamento tramite modifica in scala

Nozioni fondamentali sulla classe UIComponent

La classe UIComponent è una sottoclasse di UIObject (vedere [Classe UIComponent](#) nella *Guida di riferimento dei componenti*). È la classe base di tutti i componenti che gestiscono l'interazione con gli utenti, ovvero l'input da mouse e tastiera. La classe UIComponent consente ai componenti di effettuare le seguenti operazioni:

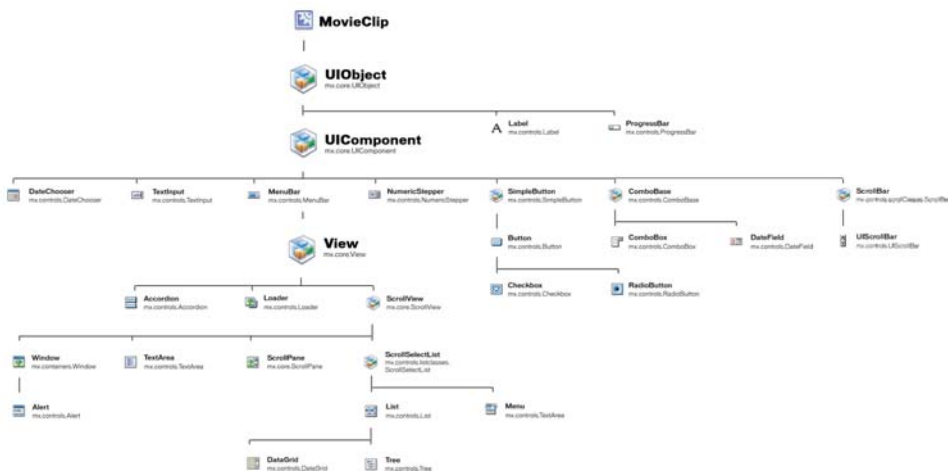
- Ricevere l'attivazione e l'input dalla tastiera
- Abilitare e disabilitare componenti
- Ridimensionare in base al layout

Estensione di altre classi della versione 2

Per facilitare la creazione dei componenti, è possibile estendere qualsiasi classe; non è pertanto indispensabile estendere direttamente le classi UIObject o UIComponent. Se si estende la classe di qualunque altro componente della versione 2 (eccetto i componenti Media), per impostazione predefinita si estendono anche le classi UIObject e UIComponent. Tutte le classi riportate nel dizionario dei componenti possono essere estese per creare una classe di componente nuova.

Ad esempio, per creare un componente con un comportamento quasi uguale al componente Button, è possibile estendere la classe Button anziché creare di nuovo tutte le funzionalità della classe Button a partire dalle classi base.

Nella figura seguente è illustrata la gerarchia dei componenti della versione 2:



Gerarchia dei componenti della versione 2

Una versione FlashPaper di questo file è disponibile nella directory di installazione di Flash nella posizione seguente: Flash 8\Samples and Tutorials\Samples\Components\arch_diagram.swf.

Informazioni sull'estensione della classe MovieClip

È possibile scegliere di non estendere una classe della versione 2 e fare in modo che il componente erediti direttamente dalla classe MovieClip di ActionScript. Se, tuttavia, si desidera utilizzare funzionalità di UIObject e UIComponent, occorre creare una classe personalizzata. Le classi UIObject e UIComponent (FirstRun/Classes/mx/core) possono essere aperte per esaminarne la struttura.

Creazione del clip filmato di un componente

Per creare un componente, è necessario creare il simbolo di un clip filmato e collegarlo al file di classe del componente.

Il clip filmato dispone di due fotogrammi e due livelli. Al primo livello, Actions, è associata una funzione globale `stop()` sul fotogramma 1. Il secondo livello, Assets, è dotato di due fotogrammi chiave. Il fotogramma 1 contiene un riquadro di delimitazione o un altro tipo di elemento grafico che verrà utilizzato come segnaposto per l'immagine finale. Il fotogramma 2 contiene tutte le altre risorse utilizzate dal componente, inclusi gli elementi grafici e le classi base.

Inserimento di un nuovo simbolo di clip filmato

Tutti i componenti sono oggetti MovieClip. Per creare un nuovo componente, è necessario inserire dapprima un nuovo simbolo in un nuovo file FLA.

Per aggiungere un nuovo simbolo di componente:

1. In Flash, creare un documento FLA vuoto.
2. Selezionare Inserisci > Nuovo simbolo.
Viene visualizzata la finestra di dialogo Crea un nuovo simbolo.
3. Immettere un nome di simbolo. Assegnare un nome al componente usando una maiuscola per la prima lettera di ogni parola che lo compone (ad esempio, MyComponent).
4. Selezionare il comportamento Clip filmato.
5. Fare clic sul pulsante Avanzate per visualizzare le impostazioni avanzate.
6. Selezionare Esporta per ActionScript e deselezionare Esporta nel primo fotogramma ed Esporta nel primo fotogramma.
7. Immettere un identificatore di concatenamento.
8. Nella casella di testo Classe AS 2.0, immettere il percorso completo della classe ActionScript 2.0.

Il nome della classe deve corrispondere al componente visualizzato nel pannello Componenti. Ad esempio, la classe del componente Button è `mx.controls.Button`.

NOTA

Non includere l'estensione del nome file; la casella di testo Classe AS 2.0 punta alla posizione della classe nel pacchetto e non al nome del file system del file.

Se il file `ActionScript` si trova in un pacchetto, includerne il nome. Il valore di questo campo può essere relativo al percorso di classe oppure può essere un percorso assoluto di pacchetto; ad esempio, `mypackage.MyComponent`.

9. Nella maggior parte dei casi, è necessario deselezionare l'opzione **Esporta** nel primo fotogramma, che risulta selezionata per impostazione predefinita. Per ulteriori informazioni, vedere [“Elenco di controllo per lo sviluppo dei componenti” a pagina 200](#).
10. Fare clic su **OK**.

Flash aggiunge il simbolo alla libreria e passa alla modalità di modifica dei simboli. In tale modalità, il nome del simbolo viene visualizzato sopra all'angolo superiore sinistro dello stage e un mirino indica il punto di registrazione del simbolo.

Modifica del clip filmato

Dopo aver creato il nuovo simbolo e averne definito i concatenamenti, è possibile definire le risorse del componente nella linea temporale del simbolo.

Il simbolo di un componente deve avere due livelli. Questa sezione descrive i livelli da inserire e gli elementi che è possibile aggiungervi.

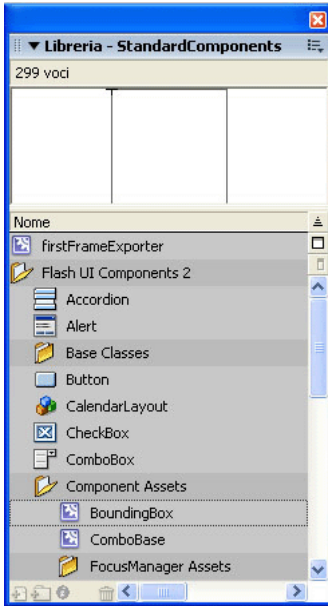
Per modificare il clip filmato:

1. Rinominare il livello 1 in **Actions** e selezionare il fotogramma 1.
2. Aprire il pannello **Azioni** e aggiungere una funzione `stop()`, come indicato di seguito:

```
stop();
```

Non aggiungere risorse di grafica in questo fotogramma.
3. Aggiungere un livello denominato **Assets**.
4. Nel livello **Assets**, selezionare il fotogramma 1 e inserire un fotogramma chiave vuoto.
In questo livello sono ora presenti due fotogrammi chiave vuoti.
5. Effettuare una delle seguenti operazioni:
 - Se il componente dispone di risorse visive che definiscono l'area di delimitazione, trascinare i simboli nel fotogramma 1 e disporli nel modo adeguato.

- Se il componente crea tutte le risorse visive in fase di runtime, trascinare un simbolo BoundingBox sullo stage nel fotogramma 1, ridimensionarlo nel modo corretto e assegnare all'istanza il nome **boundingBox_mc**. Il simbolo si trova nella libreria di StandardComponents.fla, disponibile nella cartella Configuration/ComponentFLA.



6. Se si sta creando l'estensione di un componente, inserire un'istanza del componente e di qualsiasi altra classe base nel secondo fotogramma del livello Assets.

Per effettuare questa operazione, selezionare il simbolo dal pannello Componenti e trascinarlo sullo stage. Se si estende una classe base, aprire StandardComponents.fla dalla cartella Configuration/ComponentFLA e trascinare la classe dalla libreria allo stage.

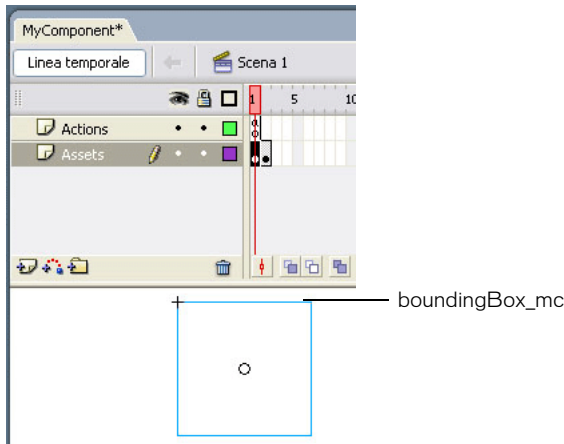
NOTA

Quando si trascina UIComponent nella libreria del componente, viene modificata la gerarchia delle cartelle all'interno della libreria. Se si prevede di riutilizzare la libreria oppure di utilizzarla con altri gruppi di componenti, ad esempio i componenti della versione 2, ristrutturare la gerarchia delle cartelle affinché corrisponda alla libreria StandardComponents.fla. In questo modo, la libreria viene organizzata nel modo corretto e si evitano simboli duplicati.

7. Aggiungere eventuali risorse di grafica utilizzate da questo componente al fotogramma 2 del livello Assets.

Qualunque risorsa utilizzata dal componente, che sia un altro componente o una risorsa multimediale come una bitmap, deve avere un'istanza inserita nel secondo fotogramma del livello Assets.

8. Il simbolo completo dovrebbe essere simile al seguente:



Definizione del clip filmato come componente

Il simbolo del clip filmato deve essere collegato al file di una classe ActionScript nella finestra di dialogo Definizione del componente. In questo modo, Flash è in grado di individuare i metatag del componente. Per ulteriori informazioni sui metatag, vedere [“Aggiunta di metadati dei componenti” a pagina 160](#). Nella finestra di dialogo Definizione dei componenti sono disponibili anche altre opzioni.

Per definire un clip filmato come componente:

1. Selezionare il clip filmato nella libreria e scegliere Definizione del componente dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse; Mac: clic tenendo premuto il tasto Control).
2. Immettere un nome di classe AS 2.0.

Se la classe fa parte di un pacchetto, specificare il nome completo del pacchetto.

3. Se necessario, utilizzare altre opzioni nella finestra di dialogo Definizione del componente:

- Fare clic sul pulsante Più (+) per definire i parametri.
Questa impostazione è opzionale. Per specificare i parametri, è consigliabile utilizzare il tag per metadatiInspectable nel file di classe del componente. Se non si specifica una classe ActionScript 2.0, definire i parametri del componente in questa area.
- Specificare un'interfaccia utente personalizzata.
Questo è il file SWF che riproduce la finestra di ispezione dei componenti. È possibile incorporarlo nel file FLA del componente oppure individuare un file SWF esterno.
- Specificare un'anteprima dal vivo.
Si tratta di un file SWF esterno o incorporato. Non è necessario specificare un'anteprima dal vivo in questo punto; è sufficiente aggiungere un riquadro di delimitazione al clip filmato del componente e l'anteprima dal vivo viene creata automaticamente. Vedere [“Creazione del clip filmato di un componente” a pagina 149](#).
- Immettere una descrizione.
Il campo Descrizione è stato considerato obsoleto in Flash MX 2004 perché il pannello Riferimento è stato rimosso. Viene fornito solo per assicurare la compatibilità con le versioni precedenti quando si salvano file FLA nel formato di Flash MX.
- Scegliere un'icona.
Questa opzione consente di specificare un file PNG, che viene utilizzato come icona per il componente. Se si specifica un tag per metadati IconFile nel file di classe ActionScript 2.0 (la scelta consigliata), questo campo viene ignorato.
- Selezionare o deselezionare "I parametri sono bloccati nelle istanze".
Se l'opzione è deselezionata, gli utenti possono aggiungere a ogni istanza del componente parametri diversi da quelli del componente. In generale, l'opzione deve essere selezionata e assicura la compatibilità con le versioni precedenti di Flash MX.
- Specificare la descrizione da visualizzare nel pannello Componenti.

Creazione del file di classe ActionScript

Tutti i simboli di componenti sono collegati a un file di classe ActionScript 2.0. Per ulteriori informazioni sul collegamento, vedere [“Creazione del clip filmato di un componente” a pagina 149](#).

Per modificare i file classe di ActionScript, è possibile usare Flash, qualsiasi editor di testo oppure qualunque IDE (Integrated Development Environment).

La classe ActionScript esterna estende un'altra classe, che si tratti di un componente della versione 2, di una classe base della versione 2 oppure della classe MovieClip di ActionScript. Estendere la classe che crea la funzionalità più simile al componente che si desidera creare. È possibile utilizzare l'ereditarietà (estensione) da una sola classe. ActionScript 2.0 non consente l'ereditarietà multipla.

Semplice esempio del file di classe di un componente

Di seguito è riportato un semplice esempio di un file di classe chiamato MyComponent.as. Se questo componente venisse creato effettivamente, sarebbe necessario collegare il file al clip filmato in Flash.

L'esempio contiene un numero ridotto di importazioni, metodi e dichiarazioni per un componente, MyComponent, che eredita dalla classe UIComponent. Il file MyComponents.as viene salvato nella cartella myPackage.

```
[Event("eventName")]

// Importa i pacchetti.
import mx.core.UIObject;

// Dichiarare la classe ed effettua l'estensione dalla classe principale.
class mypackage.MyComponent extends UIObject {

    // Identifica il nome del simbolo al quale è associata questa classe.
    static var symbolName:String = "mypackage.MyComponent";

    // Identifica il nome completo del pacchetto del proprietario del
    simbolo.
    static var symbolOwner:Object = Object(mypackage.MyComponent);

    // Fornisce la variabile className.
    var className:String = "MyComponent";

    // Definisce una funzione di costruzione vuota.
    function MyComponent() {
    }
}
```

```

// Chiama il metodo init() principale.
// Nasconde il riquadro di delimitazione; viene utilizzato
// solo durante la fase di creazione.
function init():Void {
    super.init();

    boundingBox_mc.width = 0;
    boundingBox_mc.height = 0;
    boundingBox_mc.visible = false;
}

function createChildren():Void{
    // Chiama createClassObject per creare oggetti secondari.
    size();
    invalidate();
}

function size(){
    // Scrive il codice per gestire il ridimensionamento.
    super.size();
    invalidate();
}

function draw(){
    // Scrive il codice per gestire la rappresentazione visiva.
    super.draw();
}
}

```

Panoramica del file di classe di un componente

Quella che segue è una procedura generale che illustra come creare un file ActionScript per una classe di componenti. Alcuni dei passaggi potrebbero essere opzionali, a seconda del tipo di componente da creare.

Per scrivere il file di classe di un componente:

1. (Opzionale) Importare le classi. Vedere [“Importazione delle classi” a pagina 157](#).

Questa operazione consente di fare riferimento alle classi senza specificare il pacchetto, ad esempio Button anziché mx.controls.Button.

2. Definire la classe utilizzando la parola chiave `class`; per estendere una classe principale, utilizzare la parola chiave `extend`. Vedere [“Definizione della classe e della relativa superclasse” a pagina 157](#).

3. Definire le variabili `symbolName`, `symbolOwner` e `className`. Vedere [“Identificazione dei nomi di classe, simbolo e proprietario” a pagina 157.](#)

Queste variabili sono necessarie solo nei componenti della versione 2.

4. Definire le variabili membro. Vedere [“Definizione di variabili” a pagina 159.](#)

Queste variabili possono essere utilizzate nei metodi getter/setter.

5. Definire una funzione di costruzione. Vedere [“Informazioni sulla funzione di costruzione” a pagina 176.](#)

6. Definire un metodo `init()`. Vedere [“Definizione del metodo `init\(\)`” a pagina 174.](#)

È il metodo chiamato quando viene creata la classe, se questa estende `UIComponent`. Se la classe estende `MovieClip`, questo metodo viene chiamato dalla funzione di costruzione.

7. Definire un metodo `createChildren()`. Vedere [“Definizione del metodo `createChildren\(\)`” a pagina 174.](#)

È il metodo chiamato quando viene creata la classe, se questa estende `UIComponent`. Se la classe estende `MovieClip`, questo metodo viene chiamato dalla funzione di costruzione.

8. Definire un metodo `size()`. Vedere [“Definizione del metodo `size\(\)`” a pagina 177.](#)

È il metodo chiamato quando viene ridimensionato il componente, se la classe estende `UIComponent`. Il metodo viene chiamato, inoltre, quando l'anteprima dal vivo del componente viene ridimensionata durante la creazione.

9. Definire un metodo `draw()`. Vedere [“Informazioni sulla funzione di invalidazione” a pagina 178.](#)

È il metodo chiamato quando viene invalidato il componente, se la classe estende `UIComponent`.

10. Aggiungere un tag per metadati e una dichiarazione. Vedere [“Aggiunta di metadati dei componenti” a pagina 160.](#)

L'aggiunta del tag e della dichiarazione fanno in modo che le proprietà getter/setter vengano visualizzate nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti in Flash.

11. Definire i metodi getter/setter. Vedere [“Utilizzo dei metodi getter/setter per definire i parametri” a pagina 159.](#)

12. (Opzionale) Creare le variabili per ogni elemento dello skin/concatenamento utilizzato nel componente. Vedere [“Informazioni sull'assegnazione degli skin” a pagina 181.](#)

In questo modo, gli utenti possono impostare un elemento dello skin diverso mediante la modifica di un parametro nel componente.

Importazione delle classi

Per evitare di scrivere i nomi completi delle classi in tutto il codice, è possibile importare i file delle classi. Il codice diventa, in tal modo, più breve e facilmente leggibile. Per importare una classe, utilizzare l'istruzione `import` nella parte superiore del file di classe, come indicato di seguito:

```
import mx.core.UIObject;  
import mx.core.ScrollView;  
import mx.core.ext.UIObjectExtensions;
```

```
class MyComponent extends UIComponent{
```

È anche possibile utilizzare il carattere jolly (*) per importare tutte le classi in un dato pacchetto. Ad esempio, l'istruzione seguente importa tutte le classi nel pacchetto `mx.core`:

```
import mx.core.*;
```

Se una classe importata non viene utilizzata all'interno di uno script, non viene inclusa nel risultante codice byte del file SWF. Di conseguenza, l'importazione di un intero pacchetto mediante un carattere jolly non crea un file SWF di dimensioni inutilmente grandi.

Definizione della classe e della relativa superclasse

Il file di classe di un componente viene definito come qualsiasi altro file di classe. Per indicare il nome della classe, utilizzare la parola chiave `class`. Il nome della classe deve corrispondere al nome del file di classe. Per indicare la superclasse, utilizzare la parola chiave `extends`. Per ulteriori informazioni, vedere il Capitolo 6, “Creazione di file di classi personalizzate” in *Apprendimento di ActionScript 2.0 in Flash*.

```
class MyComponentName extends UIComponent{  
  
}
```

Identificazione dei nomi di classe, simbolo e proprietario

Per consentire a Flash di individuare i pacchetti e le classi ActionScript adeguati e per mantenere la denominazione del componente, è necessario impostare le variabili `symbolName`, `symbolOwner` e `className` nel file di classe ActionScript del componente.

La variabile `symbolOwner` è un riferimento oggetto relativo a un simbolo. Se il componente corrisponde a `symbolOwner` o se la variabile `symbolOwner` è stata importata, non è necessario specificarne il nome completo.

Queste variabili sono descritte nella tabella seguente:

Variabile	Tipo	Descrizione
<code>symbolName</code>	String	Il nome della classe <code>ActionScript</code> (ad esempio, <code>ComboBox</code>). Questo nome deve corrispondere all'identificatore di concatenamento di un simbolo. La variabile deve essere di tipo statico.
<code>symbolOwner</code>	Oggetto	Il nome della classe completo (ad esempio, <code>mypackage.MyComponent</code>). Non utilizzare le virgolette per racchiudere il valore <code>symbolOwner</code> , poiché è un tipo di dati <code>Object</code> . Questo nome deve corrispondere alla classe AS 2.0 nella finestra di dialogo Proprietà del concatenamento. La variabile viene utilizzata nella chiamata interna al metodo <code>createClassObject()</code> . La variabile deve essere di tipo statico.
<code>className</code>	String	Il nome di classe del componente. Non include il nome del pacchetto e non ha un'impostazione corrispondente nell'ambiente di sviluppo di Flash. È possibile utilizzare il valore di questa variabile quando si impostano le proprietà degli stili.

Nell'esempio seguente vengono aggiunte le variabili `symbolName`, `symbolOwner` e `className` alla classe `MyButton`:

```
class MyButton extends mx.controls.Button {
    static var symbolName:String = "MyButton";
    static var symbolOwner = myPackage.MyButton;
    var className:String = "MyButton";
}
```

Definizione di variabili

Il seguente esempio di codice è contenuto nel file Button.as (mx.controls.Button). Definisce una variabile, btnOffset, da utilizzare nel file di classe, oltre alle variabili `__label` e `__labelPlacement`. Queste ultime due variabili sono precedute da un doppio carattere di sottolineatura, per evitare conflitti tra i nomi quando vengono utilizzate nei metodi getter/setter e successivamente come proprietà e parametri nel componente. Per ulteriori informazioni, vedere [“Utilizzo dei metodi getter/setter per definire i parametri” a pagina 159](#) in *Apprendimento di ActionScript 2.0 in Flash*.

```
/**
 * Numero utilizzato per l'offset dell'etichetta e/o dell'icona quando viene
 * premuto il pulsante.
 */
var btnOffset:Number = 0;

/**
 * @private
 * Testo visualizzato nell'etichetta se non si specifica un valore.
 */
var __label:String = "default value";

/**
 * @private
 * posizione predefinita dell'etichetta
 */
var __labelPlacement:String = "right";
```

Utilizzo dei metodi getter/setter per definire i parametri

Il modo più semplice per definire il parametro di un componente consiste nell'aggiungere una variabile membro pubblica che rende il parametro di tipo Inspectable: È possibile eseguire tale operazione utilizzando il tag Inspectable nel finestra di ispezione dei componenti, oppure aggiungendo la variabile Inspectable come segue:

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
```

Se tuttavia il codice che crea un componente modifica la proprietà `flavorStr`, il componente deve, generalmente, eseguire un'azione per aggiornarsi in risposta alla modifica della proprietà. Ad esempio, se `flavorStr` è impostata su "cherry", il componente potrebbe ridisegnarsi con l'immagine di una ciliegia, anziché con l'immagine predefinita di una fragola. Per le variabili membro normali, al componente non viene notificato automaticamente che il valore della variabile membro è stato modificato.

I metodi getter/setter consentono di rilevare direttamente le modifiche apportate alle proprietà di un componente. Aniché dichiarare una variabile normale con `var`, dichiarare i metodi getter/setter come illustrato di seguito:

```
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 chiamata a `invalidate()` fa in modo che il componente venga ridisegnato in base al nuovo gusto. Questo è il vantaggio che deriva dall'uso dei metodi getter/setter per la proprietà `flavorStr`, anziché di una variabile membro normale. Vedere [“Definizione del metodo draw\(\)” a pagina 177](#).

Per definire i metodi getter/setter, tenere presente quanto segue:

- È necessario anteporre `get` o `set` al nome del metodo, seguiti da uno spazio e dal nome della proprietà.
- Alla variabile che memorizza il valore della proprietà non può essere assegnato lo stesso nome del metodo getter o setter. Per convenzione, anteporre due caratteri di sottolineatura ai nomi delle variabili getter e setter.

Per definire proprietà che siano visibili nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti, vengono comunemente utilizzati i metodi getter e setter insieme ai tag.

Per ulteriori informazioni sui metodi getter/setter, vedere “Informazioni sui metodi getter e setter” in *Apprendimento di ActionScript 2.0 in Flash*.

Aggiunta di metadati dei componenti

Per fornire informazioni al compilatore sui parametri, le proprietà di associazione dei dati e gli eventi del componente, è possibile aggiungere tag per metadati del componente nei file di classe ActionScript esterni. I tag per metadati vengono utilizzati nell'ambiente di creazione di Flash per numerosi scopi.

I tag per metadati possono essere usati soltanto nei file di classe esterni di ActionScript. Non è possibile utilizzare i tag per metadati nei file FLA.

I metadati sono associati a una dichiarazione di classe oppure a un singolo campo di dati. Se il valore di un attributo è una stringa, va racchiuso tra virgolette.

Le istruzioni con metadati sono collegate alla riga successiva del file `ActionScript`. Quando si definisce una proprietà di componente, aggiungere il tag per metadati alla riga che precede la dichiarazione di proprietà. La sola eccezione è il tag per metadati `Event`. Quando si definiscono gli eventi dei componenti, aggiungere il tag per metadati fuori dalla definizione di classe, in modo che l'evento sia associato all'intera classe.

Nell'esempio seguente, i tag `Inspectable` definiscono i parametri `flavorStr`, `colorStr` e `shapeStr`:

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
[Inspectable(defaultValue="blue")]
public var colorStr:String;
[Inspectable(defaultValue="circular")]
public var shapeStr:String;
```

Nella finestra di ispezione **Proprietà** e nella scheda **Parametri** della finestra di ispezione dei componenti, tutti questi parametri vengono visualizzati come parametri di tipo `String`.

Tag per metadati

La tabella seguente descrive i tag per metadati che è possibile usare nei file di classe `ActionScript`:

Tag	Descrizione
<code>Inspectable</code>	Esponde una proprietà nella finestra di ispezione dei componenti e nella finestra di ispezione Proprietà . Vedere “Informazioni sul tag <code>Inspectable</code>” a pagina 162 .
<code>InspectableList</code>	Identifica quale sottoinsieme di proprietà modificabili deve essere elencato nella finestra di ispezione Proprietà e nella finestra di ispezione dei componenti. Se non si aggiunge un attributo <code>InspectableList</code> alla classe del componente, nella finestra di ispezione Proprietà vengono visualizzati tutti i parametri modificabili. Vedere “Informazioni sul tag <code>InspectableList</code>” a pagina 164 .
<code>Event</code>	Definisce l'evento di un componente. Vedere “Informazioni sul tag <code>Event</code>” a pagina 165 .
<code>Bindable</code>	Rivela una proprietà nella scheda Associazioni della finestra di ispezione dei componenti. Vedere “Informazioni sul tag <code>Bindable</code>” a pagina 166 .
<code>ChangeEvent</code>	Identifica un evento che causa l'associazione dei dati. Vedere “Informazioni sul tag <code>ChangeEvent</code>” a pagina 168 .

Tag	Descrizione
Collection	Identifica un attributo <code>collection</code> esposto nella finestra di ispezione dei componenti. Vedere “Informazioni sul tag Collection” a pagina 169.
IconFile	Specifica il nome file dell'icona che rappresenta il componente nel pannello Componenti. Vedere “Informazioni sul tag IconFile” a pagina 170.
ComponentTask	Specifica i nomi di file di uno o più file JSFL associati per eseguire le operazioni nell'ambiente di creazione. Vedere “Informazioni sul tag ComponentTask” a pagina 170.

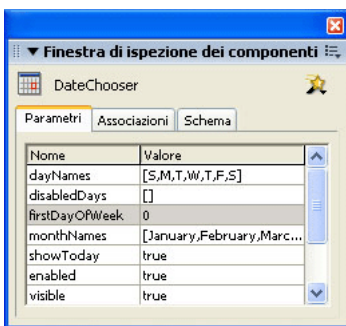
Nelle sezioni che seguono i tag per metadati dei componenti vengono descritti in maggiore dettaglio.

Informazioni sul tag Inspectable

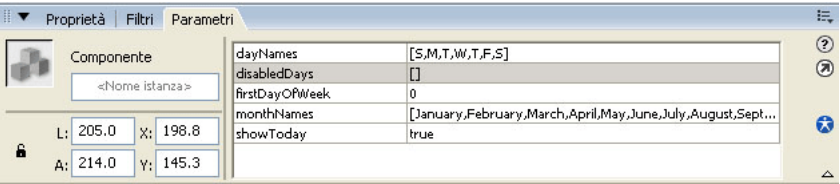
Utilizzare il tag `Inspectable` per specificare i parametri modificabili dall'utente, che vengono visualizzati nella finestra di ispezione dei componenti e nella finestra di ispezione Proprietà. Questo consente di mantenere le proprietà modificabili e il codice di `ActionScript` sottostante in una stessa posizione. Per visualizzare le proprietà dei componenti, trascinare un'istanza del componente sullo stage e selezionare la scheda Parametri nella finestra di ispezione dei componenti.

Anche i parametri di `Collection` sono modificabili. Per ulteriori informazioni, vedere [“Informazioni sul tag Collection” a pagina 169.](#)

Nella figura seguente è illustrata la scheda Parametri nella finestra di ispezione dei componenti per il componente `DateChooser`:



In alternativa, è possibile visualizzare un sottoinsieme delle proprietà del componente nella scheda Parametri della finestra di ispezione Proprietà.



Al momento di stabilire quali parametri lasciare visibili nell'ambiente di sviluppo, Flash utilizza il tag `Inspectable`. La sintassi di questo tag è la seguente:

```
[Inspectable(value_type=value[,attribute=value,...])]
property_declaration name:type;
```

L'esempio seguente definisce il parametro `enabled` come `inspectable`:

```
[Inspectable(defaultValue=true, verbose=1, category="Other")]
var enabled:Boolean;
```

Il tag `Inspectable` supporta anche attributi con tipizzazione debole, come il seguente:

```
[Inspectable("danger", 1, true, maybe)]
```

Per consentire l'associazione a una proprietà, l'istruzione di metadati deve precedere immediatamente la dichiarazione di variabile di quella proprietà.

Nella tabella seguente sono riportati gli attributi del tag `Inspectable`:

Attributo	Tipo	Descrizione
defaultValue	String o Number	(Opzionale) Un valore predefinito per la proprietà modificabile.
enumeration	String	(Opzionale) Specifica un elenco delimitato da virgole con i valori validi per la proprietà.
listOffset	Number	(Opzionale) Attributo aggiunto per garantire la compatibilità con le versioni precedenti dei componenti di Flash MX. Usato come indice predefinito in un valore List.
name	String	(Opzionale) Un nome da visualizzare per la proprietà. Ad esempio, Font Width. Se non è specificato, utilizzare il nome della proprietà, ad esempio <code>_fontWidth</code> .

Attributo	Tipo	Descrizione
<code>type</code>	String	(Opzionale) Una definizione del tipo. Se omessa, utilizzare il tipo della proprietà. I valori accettabili sono i seguenti: <ul style="list-style-type: none"> • Array • Boolean • Color • Font Name • List • Number • Oggetto • String
<code>variable</code>	String	(Opzionale) Attributo aggiunto per garantire la compatibilità con le versioni precedenti dei componenti di Flash MX. Specifica la variabile alla quale è associato questo parametro.
<code>verbose</code>	Number	(Opzionale) Una proprietà modificabile con l'attributo <code>verbose</code> impostato su 1 non viene visualizzata nella finestra di ispezione Proprietà, mentre viene visualizzata nella finestra di ispezione dei componenti. Viene utilizzata, solitamente, per le proprietà che non vengono modificate frequentemente.

Nessuno di questi attributi è obbligatorio; è possibile utilizzare `Inspectable` come tag per metadati.

Tutte le proprietà della superclasse contrassegnate `Inspectable` sono automaticamente modificabili nella classe corrente. Se si desidera nascondere alcune di queste proprietà per la classe corrente, utilizzare il tag `InspectableList`.

Informazioni sul tag `InspectableList`

Utilizzare il tag `InspectableList` per specificare quale sottoinsieme di parametri modificabili deve essere visualizzato nella finestra di ispezione Proprietà. Utilizzare `InspectableList` insieme a `Inspectable`, in modo da poter nascondere gli attributi ereditati per i componenti di tipo sottoclassi. Se non si aggiunge un tag `InspectableList` alla classe del componente, nella finestra di ispezione Proprietà vengono visualizzati tutti i parametri modificabili, inclusi quelli delle classi principali del componente.

La sintassi di `InspectableList` è la seguente:

```
[InspectableList("attribute1"[...])]
// definizione della classe
```

Il tag `InspectableList` deve immediatamente precedere la definizione della classe, poiché viene applicato all'intera classe.

L'esempio seguente consente la visualizzazione delle proprietà `flavorStr` e `colorStr` nella finestra di ispezione Proprietà, ma esclude altre proprietà modificabili dalla classe `Parent`:

```
[InspectableList("flavorStr","colorStr")]
class BlackDot extends DotParent {
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;
    [Inspectable(defaultValue="blue")]
    public var colorStr:String;
    ...
}
```

Informazioni sul tag Event

Utilizzare il tag `Event` per definire gli eventi emessi dal componente.

La sintassi di questo tag è la seguente:

```
[Event("event_name")]
```

Ad esempio, il codice seguente definisce un evento `click`:

```
[Event("click")]
```

Aggiungere le istruzioni `Event` all'esterno della definizione di classe nel file di `ActionScript`, in modo che gli eventi siano collegati alla classe, non a un membro particolare della classe.

L'esempio seguente mostra i metadati `Event` per la classe `UIObject`, che gestisce gli eventi `resize`, `move` e `draw`:

```
...
import mx.events.UIEvent;
[Event("resize")]
[Event("move")]
[Event("draw")]
class mx.core.UIObject extends MovieClip {
    ...
}
```

Per trasmettere una particolare istanza, chiamare il metodo `dispatchEvent()`. Vedere [“Uso del metodo `dispatchEvent\(\)`” a pagina 179](#).

Informazioni sul tag Bindable

L'associazione dei dati collega i componenti l'uno all'altro. L'associazione dei dati visivi si ottiene mediante la scheda Associazioni della finestra di ispezione dei componenti. In questa finestra è possibile aggiungere, visualizzare e rimuovere le associazioni per un componente.

Sebbene l'associazione dei dati funzioni con qualsiasi componente, lo scopo principale è collegare i componenti dell'interfaccia utente alle origini dati esterne, ad esempio i servizi Web e i documenti XML. Le origini dati sono disponibili come componenti con proprietà che possono essere associati ad altre proprietà di componente.

Utilizzare il tag Bindable prima di una proprietà in una classe ActionScript, per consentire la visualizzazione della proprietà nella scheda Associazione nella finestra di ispezione dei componenti. Una proprietà può essere dichiarata mediante i metodi `var` o `getter/setter`. Se una proprietà dispone sia di un metodo `getter` che di un metodo `setter`, è sufficiente applicare il tag Bindable a un solo metodo.

La sintassi del tag Bindable è la seguente:

```
[Bindable "readonly"|"writeonly",type="datatype"]
```

Entrambi gli attributi sono facoltativi.

L'esempio seguente definisce la variabile `flavorStr` come una proprietà alla quale è possibile accedere dalla scheda Associazioni della finestra di ispezione dei componenti:

```
[Bindable]  
public var flavorStr:String = "strawberry";
```

Il tag Bindable accetta tre opzioni che consentono di specificare il tipo di accesso alla proprietà, oltre al tipo di dati della proprietà. Queste opzioni sono descritte nella tabella seguente:

Opzione	Descrizione
readonly	Indica che, durante la creazione di associazioni nella finestra di ispezione dei componenti, è possibile creare solo associazioni che utilizzano questa proprietà come origine. Se, tuttavia, si utilizza ActionScript per creare associazioni, questa limitazione non è applicabile. <code>[Bindable("readonly")]</code>
writable	Indica che, durante la creazione di associazioni nella finestra di ispezione dei componenti, è possibile utilizzare questa proprietà solo come destinazione di un'associazione. Se, tuttavia, si utilizza ActionScript per creare associazioni, questa limitazione non è applicabile. <code>[Bindable("writable")]</code>
type="datatype"	Indica il tipo utilizzato dalla funzione di associazione dei dati per la proprietà. Nel resto di Flash viene utilizzato il tipo dichiarato. Se non si specifica questa opzione, l'operazione di associazione di dati utilizza il tipo di dati della proprietà definito nel codice ActionScript. Nell'esempio seguente, la funzione di associazione dei dati considera <code>x</code> come tipo <code>DataProvider</code> , anche se in realtà è un tipo <code>Object</code> : <code>[Bindable(type="DataProvider")]</code> <code>var x: Object;</code>

Tutte le proprietà di tutti i componenti possono essere incluse nell'associazione dei dati. Il tag Bindable consente semplicemente di controllare quali proprietà sono disponibili per l'associazione nella finestra di ispezione dei componenti. Se una proprietà non è preceduta dal tag Bindable, è comunque possibile utilizzarla per l'associazione dei dati, ma è necessario creare le associazioni in ActionScript.

Il tag Bindable è obbligatorio quando si utilizza il tag ChangeEvent.

Per informazioni sulla creazione di associazioni di dati nell'ambiente di creazione di Flash, vedere “Associazione dei dati (solo Flash Professional)” nella guida *Uso di Flash*.

Informazioni sul tag ChangeEvent

Il tag `ChangeEvent` consente di comunicare alla funzione di associazione dei dati che il componente emette un evento ogni volta che viene modificato il valore di una proprietà specifica. In risposta all'evento, la funzione esegue le eventuali associazioni per le quali viene utilizzata la proprietà come origine. Il componente genera l'evento solo se al suo interno è incluso il codice `ActionScript` appropriato. L'evento deve essere incluso nell'elenco dei metadati `Event` dichiarati dalla classe.

Una proprietà può essere dichiarata mediante i metodi `var` o `getter/setter`. Se una proprietà dispone sia di un metodo `getter` che di un metodo `setter`, è sufficiente applicare il tag `ChangeEvent` a un solo metodo.

La sintassi del tag `ChangeEvent` è la seguente:

```
[Bindable]
[ChangeEvent("event")]
property_declaration or getter/setter function
```

Nell'esempio seguente, il componente genera l'evento `change` quando cambia il valore della proprietà modificabile `flavorStr`:

```
[Bindable]
[ChangeEvent("change")]
public var flavorStr:String;
```

Quando si verifica l'evento specificato nei metadati, alle associazioni viene notificato che la proprietà è stata modificata.

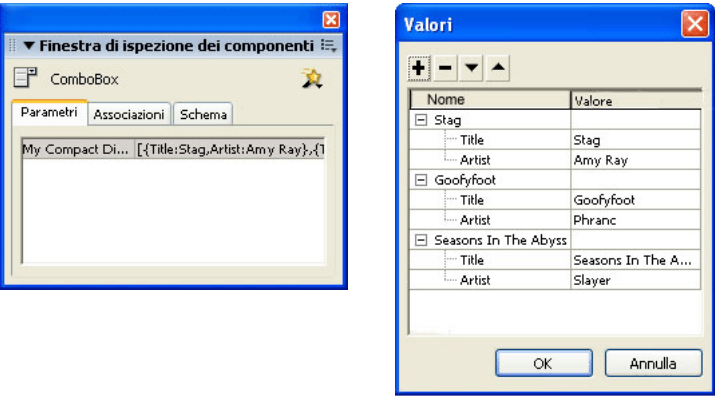
Nel tag è possibile registrare più eventi, come illustrato nell'esempio seguente:

```
[ChangeEvent("change1", "change2", "change3")]
```

Ciascuno di questi eventi indica una modifica alla proprietà. Affinché sia indicata una modifica, non è necessario che gli eventi si verifichino tutti.

Informazioni sul tag Collection

Utilizzare il tag Collection per descrivere un array di oggetti che possono essere modificati come raccolta di voci nella finestra di dialogo Valori durante la creazione. Il tipo di oggetti viene identificato mediante l'attributo `collectionItem`. Nella proprietà di una raccolta è inclusa una serie di voci della raccolta, che vengono definite dall'utente in una classe separata: la classe `mx.utils.CollectionImpl` o una sua sottoclasse. L'accesso ai singoli oggetti viene effettuato mediante i metodi della classe identificati dall'attributo `collectionClass`.



La proprietà di una raccolta nella finestra di ispezione dei componenti e la finestra di dialogo Valori che viene visualizzata quando si fa clic sulla lente di ingrandimento.

La sintassi del tag Collection è la seguente:

```
[Collection (name="name", variable="varname",
    collectionClass="mx.utils.CollectionImpl",
    collectionItem="coll-item-classname", identifier="string")]
public var varname:mx.utils.Collection;
```

Nella tabella seguente sono riportati gli attributi del tag Collection :

Attributo	Tipo	Descrizione
name	String	(Obbligatorio) Nome visualizzato nella finestra di ispezione dei componenti per la raccolta.
variable	String	(Obbligatorio) Variabile di ActionScript che fa riferimento all'oggetto Collection sottostante; ad esempio, è possibile chiamare Columns un parametro Collection e assegnare invece il nome <code>__columns</code> all'attributo <code>variable</code> sottostante.

Attributo	Tipo	Descrizione
<code>collectionClass</code>	String	(Obbligatorio) Specifica il tipo di classe di cui creare un'istanza per la proprietà della raccolta. Solitamente, è <code>mx.utils.CollectionImpl</code> , ma può anche essere una classe che estende <code>mx.utils.CollectionImpl</code> .
<code>collectionItem</code>	String	(Obbligatorio) Specifica la classe delle voci della raccolta, da memorizzare all'interno della raccolta. Questa classe include le relative proprietà modificabili, che vengono esposte dai metadati.
<code>identifier</code>	String	(Obbligatorio) Specifica il nome di una proprietà modificabile all'interno della classe delle voci della raccolta, che viene utilizzata da Flash MX come identificatore predefinito quando l'utente aggiunge una nuova voce di raccolta nella finestra di dialogo Valori. Ogni volta che l'utente crea una nuova voce di raccolta, il nome della voce viene impostato automaticamente su <i>identifier</i> più un indice univoco; ad esempio se <code>identifier=name</code> , nella finestra di dialogo Valori vengono visualizzati <code>name0</code> , <code>name1</code> , <code>name2</code> , ecc.

Per ulteriori informazioni, vedere [“Proprietà della raccolta” a pagina 201](#).

Informazioni sul tag `IconFile`

È possibile aggiungere un'icona che rappresenti il componente nel pannello Componenti dell'ambiente di creazione del codice di Flash. Per ulteriori informazioni, vedere [“Aggiunta di un'icona” a pagina 198](#).

Informazioni sul tag `ComponentTask`

È possibile specificare uno o più file Flash JavaScript (JSFL) per l'esecuzione di operazioni per il componente dall'interno dell'ambiente di creazione di Flash. Utilizzare il tag `ComponentTask` per definire questa associazione tra il componente e il relativo file JSFL e per associare eventuali file obbligatori aggiuntivi che richiedono un file JSFL. I file JSFL interagiscono con l'API JavaScript nell'ambiente di creazione di Macromedia Flash.

NOTA

Eventuali file JSFL di operazioni e file di dipendenza obbligatori dichiarati con il tag `ComponentTask` devono trovarsi nella stessa cartella del file FLA del componente quando il componente viene esportato come file SWC.

La sintassi del tag `ComponentTask` è la seguente:

```
[ComponentTask [taskName,taskFile [,otherFile[,...]]]
```

Gli attributi `taskName` e `taskFile` sono obbligatori. L'attributo `otherFile` è facoltativo.

Nell' esempio seguente `SetUp.jsfl` e `AddNewSymbol.jsfl` vengono associati alla classe dei componenti denominata `myComponent`. `AddNewSymbol.jsfl` richiede un file `testXML.xml` ed è specificato nell'attributo `otherFile`.

```
[ComponentTask("Do Some Setup","SetUp.jsfl")]
[ComponentTask("Add a new Symbol","AddNewSymbol.jsfl","testXML.xml")]
class myComponent{
    //...
}
```

Nella tabella seguente sono riportati gli attributi del tag `ComponentTask`:

Attributo	Tipo	Descrizione
<code>taskName</code>	String	(Obbligatorio) Il nome dell'operazione sotto forma di stringa. Questo nome è visualizzato nel menu a comparsa Operazioni visualizzato nella scheda Schema della finestra di ispezione dei componenti.
<code>taskFile</code>	String	(Obbligatorio) Il nome del file JSFL che implementa le operazioni all'interno dell'ambiente di creazione. Il file deve trovarsi nella stessa cartella del componente FLA quando si esporta il componente come file SWC.
<code>otherFile</code>	String	(Opzionale) Uno o più nomi di file richiesti dal file JSFL, ad esempio un file XML. Il file o i file devono trovarsi nella stessa cartella del componente FLA quando si esporta il componente come file SWC.

Definizione dei parametri dei componenti

Quando si crea un componente, è possibile aggiungere i parametri che ne definiscono l'aspetto e il comportamento. Le proprietà utilizzate più comunemente vengono visualizzate come parametri di creazione nella finestra di ispezione dei componenti e nella finestra di ispezione Proprietà. È anche possibile impostare tutti i parametri modificabili e i parametri della raccolta mediante `ActionScript`. Queste proprietà vengono definite nel file di classe del componente utilizzando il tag `Inspectable` (vedere [“Informazioni sul tag Inspectable” a pagina 162](#)).

L'esempio seguente imposta alcuni parametri di componente nel file della classe `JellyBean` e li rende visualizzabili con il tag `Inspectable` nella finestra di ispezione dei componenti:

```
class JellyBean{
    // un parametro String
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;

    // un parametro String list
```

```

Inspectable(enumeration="sour,sweet,juicy,rotten",defaultValue="sweet")
]
public var flavorType:String;

// un parametro Array
Inspectable(name="Flavors", defaultValue="strawberry,grape,orange",
verbose=1, category="Fruits")]
var flavorList:Array;

// un parametro Object
Inspectable(defaultValue="belly:flop,jelly:drop")]
public var jellyObject:Object;

// un parametro Color
Inspectable(defaultValue="#ffffff")]
public var jellyColor:Color;

// un metodo setter
Inspectable(defaultValue="default text")]
function set text(t:String)

}

```

È possibile utilizzare qualsiasi tipo di parametro tra quelli riportati di seguito:

- Array
- Oggetto
- List
- String
- Number
- Boolean
- Font Name
- Color

NOTA

La classe JellyBean è un esempio teorico. Per visualizzarne uno effettivo, vedere il file della classe Button.as, che viene installato con Flash nella directory *lingua/First Run/Classes/mx/controls*.

Informazioni sulle funzioni base

Nel file di classe del componente è necessario definire cinque funzioni: `init()`, `createChildren()`, la funzione di costruzione, `draw()` e `size()`. Quando un componente estende `UIComponent`, queste cinque funzioni nel file di classe vengono chiamate nell'ordine seguente:

- `init()`
L'eventuale inizializzazione si verifica durante la chiamata della funzione `init()`. Ad esempio, le variabili membro dell'istanza possono essere impostate in questo momento e il riquadro di delimitazione del componente può essere nascosto.
Dopo la chiamata a `init()`, vengono automaticamente impostate le proprietà `width` e `height`. Vedere [“Definizione del metodo `init\(\)`” a pagina 174](#).
- `createChildren()`
Chiamato quando un fotogramma viene riprodotto nella linea temporale. Durante questa operazione, l'utente del componente può chiamare metodi e proprietà per impostare il componente. Eventuali oggetti secondari che componente deve creare vengono creati all'interno della funzione `createChildren()`. Vedere [“Definizione del metodo `createChildren\(\)`” a pagina 174](#).
- Funzione di costruzione
Chiamata per creare un'istanza del componente. La funzione di costruzione del componente viene in genere lasciata vuota per evitare conflitti di inizializzazione. Vedere [“Informazioni sulla funzione di costruzione” a pagina 176](#).
- `draw()`
Eventuali elementi visivi del componente creati o modificati a livello di codice dovrebbero verificarsi all'interno della funzione di disegno. Vedere [“Definizione del metodo `draw\(\)`” a pagina 177](#).
- `size()`
Questa funzione viene chiamata ogni volta che componente viene ridimensionato in fase di runtime e le vengono passate le proprietà `width` e `height` aggiornate del componente. Gli oggetti secondari del componente possono essere ridimensionati o spostati rispetto alle proprietà `width` e `height` aggiornate del componente all'interno della funzione `size()`. Vedere [“Definizione del metodo `size\(\)`” a pagina 177](#).

Queste funzioni del componente di base sono illustrate in dettaglio nelle sezioni seguenti.

Definizione del metodo `init()`

Il metodo `init()` viene chiamato in Flash quando viene creata la classe. Il metodo viene chiamato una sola volta durante la creazione dell'istanza del componente.

Utilizzare il metodo `init()` per effettuare le seguenti operazioni:

- Chiamare `super.init()`.
Questa operazione è obbligatoria.
- Rendere invisibile `boundingBox_mc`.

```
boundingBox_mc.width = 0;  
boundingBox_mc.height = 0;  
boundingBox_mc.visible = false;
```
- Creare variabili membro dell'istanza.

I parametri `width`, `height` e `clip` vengono impostati correttamente solo dopo che è stato chiamato questo metodo.

Il metodo `init()` viene chiamato dalla funzione di costruzione di `UIObject`, in modo che il flusso di controllo risalga la catena delle funzioni di costruzione finché non raggiunge `UIObject`. La funzione di costruzione di `UIObject` chiama il metodo `init()` definito nella sottoclasse inferiore. Per consentire alla classe base di completare l'inizializzazione, ogni implementazione di `init()` deve chiamare `super.init()`. Se si implementa un metodo `init()` e non si chiama `super.init()`, il metodo `()init` non viene chiamato su alcuna classe base che, di conseguenza, potrebbe non essere mai in uno stato utilizzabile.

Definizione del metodo `createChildren()`

I componenti implementano il metodo `createChildren()` per creare oggetti secondari (come altri componenti) all'interno del componente. Per creare un'istanza di un oggetto secondario del componente, anziché chiamare la funzione di costruzione dell'oggetto secondario nel metodo `createChildren()`, chiamare `createClassObject()` o `createObject()`.

È consigliabile chiamare `size()` all'interno del metodo `createChildren()`, per assicurarsi che tutti gli elementi secondari vengano inizialmente impostati sulla dimensione corretta. Inoltre, per aggiornare lo schermo, chiamare `invalidate()` all'interno del metodo `createChildren()`. Per ulteriori informazioni, vedere [“Informazioni sulla funzione di invalidazione” a pagina 178](#).

Di seguito è riportata la sintassi del metodo `createClassObject()`:

```
createClassObject(className, instanceName, depth, initObject)
```

Nella seguente tabella sono descritti i parametri:

Parametro	Tipo	Descrizione
<i>className</i>	Oggetto	Il nome di classe.
<i>instanceName</i>	String	Il nome dell'istanza.
<i>depth</i>	Number	La profondità dell'istanza.
<i>initObject</i>	Oggetto	Un oggetto che contiene le proprietà di inizializzazione.

Per chiamare `createClassObject()`, è necessario sapere quali sono gli elementi secondari, in quanto nella chiamata a `createClassObject()` devono essere specificati il nome e il tipo dell'oggetto, oltre agli eventuali parametri di inizializzazione.

Nell'esempio seguente, viene chiamato `createClassObject()` per creare un nuovo oggetto `Button` da utilizzare all'interno del componente:

```
up_mc.createClassObject(mx.controls.Button, "submit_btn", 1);
```

Per impostare le proprietà nella chiamata a `createClassObject()`, aggiungerle con il parametro *initObject*. L'esempio seguente imposta il valore della proprietà `label`:

```
form.createClassObject(mx.controls.CheckBox, "cb", 0, {label:"Check  
this"});
```

L'esempio seguente crea i componenti `TextInput` and `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()  
}
```

Informazioni sulla funzione di costruzione

Le funzioni di costruzione sono riconoscibili, in quanto hanno lo stesso nome della classe del componente. Ad esempio, nel codice seguente viene illustrata la funzione di costruzione del componente `ScrollBar`:

```
function ScrollBar() {  
}
```

In questo caso, quando viene creata un'istanza di una nuova barra di scorrimento, viene chiamata la funzione di costruzione `ScrollBar()`.

In generale, le funzioni di costruzione dei componenti devono essere vuote. L'impostazione delle proprietà nelle funzioni di creazione può portare, a volte, alla sovrascrittura dei valori predefiniti, a seconda dell'ordine delle chiamate di inizializzazione.

Se il componente estende `UIComponent` o `UIObject`, vengono chiamati automaticamente i metodi `init()`, `createChildren()` e `size()`, ed è possibile lasciare vuota la funzione di costruzione, come illustrato di seguito:

```
class MyComponent extends UIComponent{  
    ...  
    // questa è la funzione di costruzione  
    function MyComponent(){  
    }  
}
```

Per tutti i componenti della versione 2 dovrebbe essere definita una funzione `init()`, che viene chiamata dopo la chiamata alla funzione di costruzione. Inserire il codice di inizializzazione nella funzione `init()` del componente. Per ulteriori informazioni, consultare la sezione successiva.

Se il componente estende `MovieClip`, si consiglia di chiamare un metodo `init()`, un metodo `createChildren()` e un metodo che crei il layout del componente dalla funzione di costruzione, come illustrato nell'esempio di codice seguente:

```
class MyComponent extends MovieClip{  
    ...  
    function MyComponent(){  
        init()  
    }  
  
    function init():Void{  
        createChildren();  
        layout();  
    }  
    ...  
}
```


Per ulteriori informazioni sulle funzioni di costruzione, vedere “Creazione della funzione di costruzione” in *Apprendimento di ActionScript 2.0 in Flash*.

Definizione del metodo draw()

È possibile scrivere codice nel metodo `draw()` per creare o modificare gli elementi visivi di un componente. In altre parole, nel metodo `draw()` un componente viene disegnato automaticamente in base alle variabili di stato. Dall'ultima chiamata al metodo `draw()`, è possibile che siano stati chiamati più metodi o proprietà. Tenerne conto nel corpo di `draw()`.

Evitare, tuttavia, di chiamare direttamente il metodo `draw()`. Chiamare invece il metodo `invalidate()`, in modo che sia possibile inserire in coda e gestire in batch le chiamate a `draw()`. Questo approccio consente di migliorare l'efficienza e centralizzare il codice. Per ulteriori informazioni, vedere “[Informazioni sulla funzione di invalidazione](#)” a pagina 178.

All'interno del metodo `draw()`, è possibile utilizzare chiamate all'API di disegno di Flash per disegnare bordi, righelli e altri elementi grafici. È anche possibile impostare i valori delle proprietà e i metodi di chiamata. Per rimuovere gli oggetti visibili, può essere chiamato il metodo `clear()`.

Nell'esempio seguente, relativo al componente Dial (vedere “[Creazione del primo componente](#)” a pagina 135), il metodo `draw()` imposta la rotazione della lancetta in base alla proprietà `value`:

```
function draw():Void {  
    super.draw();  
    dial.needle._rotation = value;  
}
```

Definizione del metodo size()

Quando si ridimensiona un componente in runtime utilizzando il metodo `componentInstance.setSize()`, viene chiamata la funzione `size()` e vengono passate le proprietà `width` e `height`. Per posizionare il contenuto del componente, è possibile utilizzare il metodo `size()` nel file di classe del componente.

Come minimo, il metodo `size()` dovrebbe chiamare il metodo `size()` della superclasse (`super.size()`).

Nell'esempio seguente, dal componente Dial (vedere [“Creazione del primo componente” a pagina 135](#)), il metodo `size()` utilizza i parametri `width` e `height` per ridimensionare il clip filmato del quadrante:

```
function size():Void {  
    super.size();  
    dial._width = width;  
    dial._height = height;  
    invalidate();  
}
```

Chiamare il metodo `invalidate()` all'interno del metodo `size()` per assegnare tag di ridisegno al componente, anziché chiamare direttamente il metodo `draw()`. Per ulteriori informazioni, vedere la sezione seguente.

Informazioni sulla funzione di invalidazione

Macromedia consiglia di evitare, nella maggior parte dei casi, che un componente si aggiorni immediatamente, ma di fare in modo che venga salvata una copia del nuovo valore della proprietà, che venga impostato un flag per indicare le modifiche e che quindi venga chiamato il metodo `invalidate()`. Questo metodo indica che sono stati modificati solo gli elementi visivi dell'oggetto, mentre le dimensioni e la posizione degli oggetti secondari sono rimaste invariate. Questo metodo chiama il metodo `draw()`.

Durante la creazione dell'istanza di un componente, è necessario chiamare un metodo di invalidazione almeno una volta. Solitamente, questa operazione viene effettuata nel metodo `createChildren()` o `layoutChildren()`.

Invio degli eventi

Se si desidera che il componente trasmetta altri eventi, oltre a quelli che eredita dalla classe principale, chiamare il metodo `dispatchEvent()` nel file di classe del componente.

Il metodo `dispatchEvent()` viene definito nella classe `mx.events.EventDispatcher` e viene ereditato da tutti i componenti che estendono `UIObject`. Vedere [“Classe EventDispatcher”](#) nella *Guida di riferimento dei componenti*.

Per ogni nuovo evento, è inoltre necessario aggiungere un tag per metadati Event nella parte superiore del file di classe. Per ulteriori informazioni, vedere [“Informazioni sul tag Event” a pagina 165](#).

NOTA

Per informazioni sulla gestione degli eventi dei componenti in un'applicazione Flash, consultare il [Capitolo 4, “Gestione degli eventi dei componenti” a pagina 69](#).

Uso del metodo `dispatchEvent()`

Nel corpo del file di classe `ActionScript` del componente, gli eventi vengono inviati tramite il metodo `dispatchEvent()`. Di seguito è riportata la sintassi del metodo `dispatchEvent()`:

```
dispatchEvent(eventObj)
```

Il parametro `eventObj` è un oggetto `ActionScript` che descrive l'evento (vedere l'esempio di seguito in questa sezione).

Prima di chiamarlo, dichiarare il metodo `dispatchEvent()` nel codice, come riportato di seguito:

```
private var dispatchEvent:Function;
```

È inoltre necessario creare un oggetto evento da passare a `dispatchEvent()`. Nell'oggetto evento sono contenute informazioni sull'evento, che possono essere utilizzate dal listener per gestire l'evento stesso.

È possibile creare un oggetto evento esplicitamente prima di inviare l'evento, come illustrato nell'esempio seguente:

```
var eventObj = new Object();  
eventObj.type = "myEvent";  
eventObj.target = this;  
dispatchEvent(eventObj);
```

È inoltre possibile utilizzare una sintassi abbreviata in un'unica riga, che imposti il valore delle proprietà `type` e `target` ed esegua l'invio dell'evento:

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

Nell'esempio precedente, l'impostazione della proprietà `target` è opzionale, poiché è implicita.

La descrizione di ogni evento nella documentazione di Flash 8 riporta le proprietà opzionali e quelle obbligatorie. Ad esempio, l'evento `ScrollBar.scroll` richiede una proprietà `detail`, oltre a `type` e `target`. Per ulteriori informazioni, vedere le descrizioni degli eventi nella *Guida di riferimento dei componenti*.

Eventi comuni

Nella tabella seguente sono elencati gli eventi comuni che vengono trasmessi da diverse classi. Ogni componente dovrebbe trasmettere questi eventi, qualora abbiano rilevanza nel contesto del componente stesso. Non si tratta di un elenco completo di eventi per tutti i componenti, ma contiene soltanto quelli che hanno maggiore probabilità di essere riutilizzati da altri componenti. Sebbene per alcuni eventi non siano specificati parametri, tutti gli eventi hanno un parametro implicito: un riferimento all'oggetto che trasmette l'evento.

Evento	Uso
<code>click</code>	Utilizzato dal componente <code>Button</code> o tutte le volte che un clic del mouse non ha altro significato.
<code>change</code>	Utilizzato da <code>List</code> , <code>ComboBox</code> e altri componenti per l'immissione del testo.
<code>scroll</code>	Utilizzato da <code>ScrollBar</code> e da altri controlli che provocano lo scorrimento (i pulsanti di scorrimento in un menu a comparsa scorrevole).

Inoltre, a causa dell'ereditarietà dalle classi base, tutti i componenti trasmettono gli eventi riportati di seguito:

Evento UIComponent	Descrizione
<code>load</code>	Il componente sta creando o caricando gli oggetti secondari.
<code>unload</code>	Il componente sta scaricando gli oggetti secondari.
<code>focusIn</code>	Il componente è attivo. Anche alcuni componenti equivalenti di HTML (<code>ListBox</code> , <code>ComboBox</code> , <code>Button</code> , <code>Text</code>) potrebbero trasmettere <code>focus</code> , ma tutti emettono <code>DOMFocusIn</code> .
<code>focusOut</code>	Il componente non è più attivo.
<code>move</code>	Il componente è stato spostato in un'altra posizione.
<code>resize</code>	Il componente è stato ridimensionato.

La tabella seguente riporta gli eventi di tastiera più comuni:

Eventi di tastiera	Descrizione
<code>keyDown</code>	Viene premuto un tasto. La proprietà <code>code</code> contiene il codice tasto e la proprietà <code>ascii</code> contiene il codice ASCII del tasto premuto. Non controllare l'oggetto <code>Key</code> di basso livello, poiché l'evento potrebbe non essere stato generato da tale oggetto.
<code>keyUp</code>	Viene rilasciato un tasto.

Informazioni sull'assegnazione degli skin

Un componente dell'interfaccia utente (UI), è composto interamente da clip filmato associati. Questo significa che tutte le risorse di un componente dell'interfaccia utente possono essere esterne al clip filmato del componente stesso, in modo da poter essere utilizzate da altri componenti. Ad esempio, se il componente necessita di funzionalità per le caselle di controllo, è possibile riutilizzare le risorse esistenti del componente CheckBox.

Il componente CheckBox utilizza un clip filmato separato per rappresentare ognuno dei suoi stati, quali FalseUp, FalseDown, Disabled, Selected e così via. Tuttavia, a questi stati è possibile associare dei clip filmato personalizzati, detti *skin*. In fase di esecuzione, i clip filmato vecchi e nuovi vengono esportati nel file SWF. Gli stati precedenti diventano semplicemente invisibili per lasciare il posto ai nuovi clip filmato. Questa possibilità di cambiare skin sia durante la fase di creazione che in runtime, viene detta *associazione di skin*.

Per associare skin ai componenti, creare una variabile per ogni elemento dello skin (il simbolo del clip filmato) utilizzato nel componente, quindi impostarlo sull'identificatore di concatenamento del simbolo. In questo modo, lo sviluppatore è in grado di impostare un elemento dello skin diverso con la semplice modifica di un parametro nel componente, come illustrato di seguito:

```
var falseUpIcon = "mySkin";
```

L'esempio seguente mostra le variabili di skin per i vari stati 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";
```

Informazioni sugli stili

Gli stili possono essere utilizzati per la registrazione di tutti gli elementi grafici contenuti nel componente in una classe, in modo da consentire successivamente a tale classe di controllare la combinazione di colori degli elementi grafici in runtime. Per supportare gli stili non è necessario nessun codice particolare nelle implementazioni dei componenti. Gli stili sono implementati interamente negli skin e nelle classi base (UIObject e UICComponent) .

Per aggiungere un nuovo stile a un componente, chiamare `getStyle("styleName")` nella classe del componente. Se lo stile è stato impostato su un'istanza, su un foglio di stile personalizzato oppure sul foglio di stile globale, viene recuperato il valore. In caso contrario, potrebbe essere necessario installare un valore predefinito dello stile nel foglio di stile globale. Per ulteriori informazioni sugli stili, vedere [“Uso degli stili per personalizzare il testo e il colore dei componenti” a pagina 86](#).

Registrazione degli skin negli stili

Nell'esempio seguente viene creato un componente denominato Shape. Questo componente visualizza una forma corrispondente a uno dei due skin: un cerchio o un quadrato. Gli skin vengono registrati nello stile `themeColor`.

Per registrare uno skin in uno stile:

1. Creare un nuovo file `ActionScript` e copiarvi il seguente codice:

```
import mx.core.UICComponent;

class Shape extends UICComponent{

    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. Salvare il file con il nome Shape.as.
3. Creare un nuovo documento Flash e salvarlo come Shape.fla nella stessa cartella in cui si trova Shape.as.
4. Disegnare un cerchio sullo stage, selezionarlo e premere F8 per convertirlo in un clip filmato.
Assegnare al cerchio il nome e l'identificatore di concatenamento **circle_skin**.
5. Aprire il clip filmato `circle_skin` e inserire il seguente codice ActionScript nel fotogramma 1 per registrare il simbolo con il nome di stile `themeColor`:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");
```
6. Creare un nuovo clip filmato per il componente.
Denominare il clip filmato e l'identificatore di concatenamento **Shape**.
7. Creare due livelli. Inserire una funzione `stop()` nel primo fotogramma del primo livello. Inserire il simbolo `circle_skin` nel secondo fotogramma.
Questo è il clip filmato del componente. Per ulteriori informazioni, vedere [“Creazione del clip filmato di un componente” a pagina 149](#).
8. Aprire `StandardComponents.fla` come una libreria esterna, quindi trascinare il clip filmato `UIComponent` sullo stage nel secondo fotogramma del clip filmato `Shape` (con `circle_skin`).
9. Chiudere `StandardComponents.fla`.
10. Selezionare il clip filmato `Shape` nella libreria e scegliere Definizione del componente dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control), e immettere il nome della classe AS 2.0 **Shape**.
11. Provare il clip filmato con il componente `Shape` sullo stage.
Per cambiare il colore del tema, impostare lo stile sull'istanza. Il codice seguente consente di cambiare in rosso il colore del componente `Shape` con il nome di istanza `shape`:

```
shape.setStyle("themeColor",0xff0000);
```
12. Disegnare un quadrato sullo stage e convertirlo in un clip filmato.
Immettere il nome di concatenamento **square_skin** e assicurarsi che l'opzione Esporta nel primo fotogramma sia selezionata.

NOTA

Poiché il clip filmato non è inserito nel componente, è necessario selezionare Esporta nel primo fotogramma per rendere disponibile lo skin prima dell'inizializzazione.

- 13.** Aprire il clip filmato `square_skin` e inserire il seguente codice ActionScript nel fotogramma 1 per registrare il simbolo con il nome di stile `themeColor`:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");
```

- 14.** Inserire il codice seguente nell'istanza del componente Shape sullo stage, nella linea temporale principale:

```
onClipEvent(initialize){  
    themeShape = "square_skin";  
}
```

- 15.** Provare il clip filmato con il componente Shape sullo stage. Dovrebbe essere visualizzato un quadrato rosso.

Registrazione di un nuovo nome di stile

Se è stato creato un nuovo nome di stile di tipo colore, aggiungere il nuovo nome all'oggetto `colorStyles` nel file `StyleManager.as` (First Run\Classes\mx\styles\StyleManager.as). Questo esempio aggiunge lo stile `shapeColor`:

```
// inizializza l'insieme degli stili di colore che ereditano  
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  
};
```


Registrare il nuovo nome di stile negli skin del cerchio e del quadrato sul fotogramma 1 di ogni clip filmato dello skin, come illustrato di seguito:

```
mx.skins.ColoredSkinElement.setColorStyle(this, "shapeColor");
```

È possibile cambiare il colore con il nuovo nome di stile, impostando lo stile nell'istanza, come illustrato di seguito:

```
shape.setStyle("shapeColor",0x00ff00);
```

Come incorporare componenti esistenti in un componente

In questa sezione, si apprenderà come creare un semplice componente LogIn che incorpora i componenti Label, TextInput e Button. In questa esercitazione verrà dimostrato come vengono incorporati i componenti esistenti in nuovi componenti, aggiungendo i relativi simboli della libreria Flash (FLA) non compilati. I file del componente completi, LogIn fla, LogIn.as e LogIn.swf, si trovano nella cartella degli esempi sul disco rigido:

- In Windows: cartella C:\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Login.
- In Macintosh: cartella HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Login.

Il componente LogIn fornisce un'interfaccia per l'immissione del nome e della password. L'API per LogIn dispone di due proprietà, `name` e `password`, per impostare e ottenere i valori stringa nei campi di TextInput relativi al nome e alla password. Quando l'utente fa clic su un pulsante con etichetta "LogIn" (Accesso), il componente LogIn invia inoltre un evento "click".

- ["Creazione del file Flash \(FLA\) LogIn" a pagina 185](#)
- ["Il file di classe LogIn" a pagina 188](#)
- ["Prova ed esportazione del componente LogIn" a pagina 192](#)

Creazione del file Flash (FLA) LogIn

Creare innanzitutto un file Flash (FLA) in cui verrà inserito il simbolo del componente.

Per creare il file FLA LogIn:

1. In Flash, selezionare File > Nuovo e creare un nuovo documento.
2. Scegliere File > Salva con nome e salvare il file come LogIn fla.
3. Selezionare Inserisci > Nuovo simbolo. Denominarlo **Login** e selezionare il pulsante di opzione tipo di clip Filmato.

Se la sezione Concatenamento della finestra di dialogo Crea un nuovo simbolo non è visibile, fare clic sul pulsante Avanzato per aprirla.

4. Selezionare Esporta per ActionScript e deselezionare Esporta nel primo fotogramma.
5. Immettere un identificatore di concatenamento.

L'identificatore di concatenamento predefinito è LogIn. Nei punti successivi di questa procedura verrà utilizzato il valore predefinito.

6. Immettere **LogIn** nella casella di testo Classe ActionScript 2.0. Questo valore è il nome di classe del componente.

Se la classe è inclusa in un pacchetto, specificare il nome completo del pacchetto. Ad esempio, mx.controls.CheckBox indica la classe CheckBox nel pacchetto mx.controls.

7. Fare clic su OK.

Viene aperta automaticamente la modalità di modifica dei simboli.

8. Inserire un nuovo livello. Assegnare al livello superiore il nome **Actions** e al livello inferiore il nome **Assets**.

9. Selezionare il fotogramma 2 nel livello Assets e inserire un fotogramma chiave (F6).

Questa è la struttura del clip filmato del componente: un livello Actions e un livello Assets. Nel livello Actions è presente un fotogramma chiave; nel livello Assets sono presenti 2 fotogrammi chiave.

10. Selezionare il fotogramma 1 nel livello Actions e aprire il pannello Azioni (F9). Immettere una funzione globale `stop()`.

In questo modo si impedisce al clip filmato di passare al fotogramma 2.

11. Scegliere File > Importa > Apri libreria esterna e selezionare il file StandardComponents.fla dalla cartella Configuration/ComponentFLA.

- In Windows: \Programmi\Macromedia\Flex 8\lingua\Configuration\ComponentFLA\StandardComponents.fla.
- In Macintosh: HD/Applicazioni/Macromedia Flex 8/Configuration/ComponentFLA/StandardComponents.fla

NOTA

Per informazioni sui percorsi delle cartelle, vedere "Cartelle di configurazione installate con Flex" in Uso di Flex.

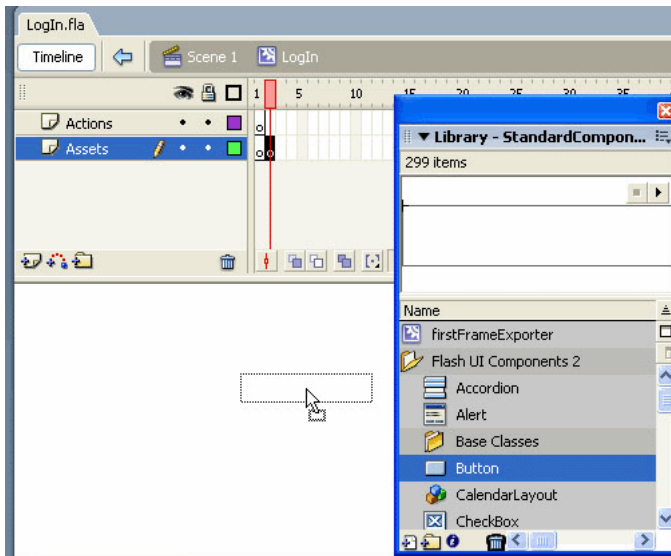
12. Selezionare il fotogramma 2 nel livello Assets. Dalla libreria StandardComponents.fla, passare alla cartella Flex UI Components 2. Trascinare un simbolo di componente Button, Label e TextInput nel fotogramma 2 del livello Assets.

Le dipendenze di risorse per questi componenti vengono copiate automaticamente nella libreria LogIn.fla.

Tutte le risorse del componente vengono aggiunte al fotogramma 2 del livello Assets. Poiché sul fotogramma 1 del livello Actions è presente una funzione globale `stop()`, le risorse nel fotogramma 2 non sono visibili nel modo in cui sono disposte sullo stage.

Le risorse vengono aggiunte al fotogramma 2 per due motivi:

- Per copiare automaticamente nella libreria tutte le risorse e le risorse secondarie e renderle disponibili per crearne un'istanza dinamicamente e per accedere ai metodi, alle proprietà e agli eventi delle risorse stesse.
- Per fare in modo che mediante l'inserimento in un fotogramma le risorse vengano caricate in maniera più fluida durante lo streaming del filmato, evitando di impostare l'esportazione delle risorse nella libreria prima del primo fotogramma. Questo metodo impedisce che si verifichi un picco di elaborazione iniziale durante il trasferimento dei dati, che potrebbe determinare ritardi o pause prolungate durante lo scaricamento.



Trascinamento di un simbolo di componente Button dalla libreria in StandardComponents.fla nel fotogramma 2 del livello Assets di LogIn.fla

13. Chiudere la libreria StandardComponents.fla.
14. Nel livello Assets, selezionare il fotogramma 1. Trascinare il clip filmato BoundingBox dalla libreria LogIn.fla (cartella Components Assets) sullo stage.
15. Assegnare all'istanza BoundingBox il nome **boundingBox_mc**.

16. Utilizzare il pannello Informazioni per ridimensionare BoundingBox in base alle dimensioni del clip filmato LogInFinal (340, 150) e impostarne la posizione su 0, 0.

L'istanza BoundingBox viene utilizzata per creare l'anteprima dal vivo del componente e per consentire all'utente di gestirne il ridimensionamento durante la creazione. Le dimensioni del riquadro di delimitazione devono essere tali da poter racchiudere tutti gli elementi grafici presenti nel componente.

NOTA

Se si estende un componente, incluso qualsiasi componente della versione 2, è necessario mantenere i nomi di istanze già utilizzati da quel componente, in quanto il codice fa riferimento ad essi. Se, ad esempio, si include un componente della versione 2 che utilizza già il nome di istanza boundingBox_mc, evitare di modificare il nome. Per i nomi di istanza personalizzati, è possibile utilizzare qualsiasi nome univoco che non crei conflitti con un nome già esistente nella stessa area di validità.

17. Selezionare il clip filmato LogIn nella libreria e scegliere Definizione del componente dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control).
18. Nella casella di testo Classe AS 2.0, immettere **LogIn**.
- Questo valore è il nome di classe ActionScript. Se la classe è inclusa in un pacchetto, il valore è il nome completo del pacchetto. Ad esempio, mx.controls.CheckBox indica la classe CheckBox nel pacchetto mx.controls.
19. Fare clic su OK.
20. Salvare il file.

Il file di classe LogIn

Di seguito è riportata la classe ActionScript per il componente LogIn. Per una descrizione di ogni sezione, leggere i commenti all'interno del codice stesso. Per informazioni dettagliate sugli elementi di un file di classe di un componente, vedere [“Panoramica del file di classe di un componente” a pagina 155](#).

Per creare questo file, aprire un nuovo file ActionScript in Flash oppure utilizzare un altro editor di testo. Salvare il file con il nome LogIn.as nella stessa cartella che contiene il file LogIn.fla.

È possibile copiare o digitare nel nuovo file LogIn.as il seguente codice di classe ActionScript del componente LogIn. Se si digita il codice, invece di copiarlo, è più facile acquisire velocemente dimestichezza con ogni elemento del codice del componente.

```
/* Importa i pacchetti in modo che sia possibile fare riferimento a essi
   direttamente dalla classe. */
import mx.core.UIComponent;
```

```

import mx.controls.Label;
import mx.controls.TextInput;
import mx.controls.Button;

// Tag per metadati Event
[Event("change")]
[Event("click")]
class LogIn extends UIComponent
{
    /* I componenti devono dichiarare queste variabili di membro per essere
    correttamente
    rappresentati nell'architettura dei componenti. */
    static var symbolName:String = "LogIn";
    static var symbolOwner:Object = LogIn;
    var className:String = "LogIn";

    // Rappresentazione grafica 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;

    /* La variabile del membro privato "__value" è pubblicamente disponibile
    tramite getter/setter.
    Questi rappresentano i valori di stringa InputText relativi a nome e
    password. */
    private var __name:String;
    private var __password:String;

    /* Funzione di costruzione:
    Sebbene richiesti per tutte le classi, i componenti della versione 2
    richiedono
    che la funzione di costruzione sia vuota, con zero argomenti.
    Tutte le operazioni di inizializzazione vengono eseguite in un metodo
    init
    obbligatorio dopo la costruzione dell'istanza della classe. */
    function LogIn() {
    }

    /* Codice di inizializzazione:
    Il metodo init è obbligatorio per i componenti della versione 2. Deve
    inoltre chiamare
    a sua volta il metodo init() della classe principale con super.init().
    Il metodo init è obbligatorio per i componenti che estendono
    UIComponent. */
    function init():Void {
        super.init();
    }
}

```

```

        boundingBox_mc._visible = false;
        boundingBox_mc._width = 0;
        boundingBox_mc._height = 0;
    }

    /* Crea gli oggetti secondari richiesti all'avvio:
       Il metodo createChildren è obbligatorio per i componenti
       che estendono 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;
        /* Impostare questo campo di testo di input in modo che sia attivo.
        Nota: assicurarsi di selezionare Controllo > Disattiva scelte rapide da
        tastiera.
        in Flash Debugger se non è già selezionato, altrimenti
        potrebbe non venire attivato durante la prova. */
        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();
    }

    /* Il metodo draw è obbligatorio per i componenti della versione 2.
       Viene chiamato dopo che il componente è

```

```

        stato invalidato mediante una chiamata a invalidate().
        Include le modifiche in un'operazione batch di ridisegno,
        anziché effettuarle individualmente. Questo approccio consente
        una maggiore efficienza e una migliore centralizzazione del codice. */
function draw():Void {
    super.draw();
}

/* Il metodo size viene richiamato quando le dimensioni del componente
    cambiano. A questo punto, è possibile ridimensionare gli elementi
    secondari
    Il metodo size è obbligatorio per i componenti che estendono
    UIComponent. */
function size():Void {
    super.size();
    // Causa un ridisegno, se necessario.
/   invalidate();
}

/* Gestore di eventi:
    Chiamato dal pulsante LogIn quando riceve un clic del mouse.
    Poiché si desidera che questo evento sia accessibile al di fuori
    dell'area di validità di
    questo componente, l'evento click viene inviato tramite dispatchEvent.
    */
public function click(evt){
    // Aggiorna le variabili di membro con il contenuto del campo di input.
    __name = name_ti.text;
    __password = password_ti.text;
    // Invia un evento click quando si fa clic sul pulsante.
    dispatchEvent({type:"click"});
}

/* Questa è la funzione getter/setter della proprietà name.
    I metadati [Inspectable] consentono la visualizzazione della proprietà
    nella finestra di ispezione Proprietà e l'impostazione di un
    valore predefinito. Se si utilizza una funzione getter/setter, è
    possibile chiamare invalidate
    e imporre il ridisegno da parte del componente quando viene modificato
    il valore. */
[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);
    }

}

```

Prova ed esportazione del componente LogIn

È stato creato il file Flash contenente gli elementi grafici, le classi base e il file di classe che comprendono tutte le funzionalità del componente LogIn. A questo punto, è possibile provare il componente.

Idealmente, la prova del componente dovrebbe essere effettuata durante la creazione, specialmente durante la scrittura del file di classe. Il modo più veloce per effettuare una prova consiste nel convertire il componente in un clip compilato e utilizzarlo nel file FLA del componente.

Dopo aver completato la creazione di un componente, esportarlo in un file SWC. Per ulteriori informazioni, vedere [“Esportazione e distribuzione di un componente” a pagina 194](#).

Per provare il componente LogIn:

1. nel file LogIn.fla, selezionare il clip filmato LogIn nella libreria e scegliere Conversione in clip compilato dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control).

Alla libreria viene aggiunto un clip compilato, denominato LogIn SWF. La compilazione viene eseguita a solo scopo di prova. In caso contrario, dovrebbero venire applicate le istruzioni indicate più avanti in questa sezione per esportare il clip filmato LogIn.

NOTA

Se è già stato creato un clip compilato, ad esempio se è la seconda o la terza volta che si esegue la prova, viene visualizzata la finestra di dialogo Risolvi il conflitto tra librerie. Scegliere Sostituisci gli elementi esistenti per aggiungere la nuova versione al documento.

2. Trascinare login SWF nello stage nel fotogramma 1 della linea temporale principale (assicurandosi di trovarsi nella linea temporale principale, Scena 1, e non nella linea temporale del clip filmato).

È possibile impostare la proprietà relativa al nome e alla password nella scheda Parametri o nella finestra di ispezione dei componenti. Questo è utile se si desidera che un testo predefinito, come "Inserire il proprio nome" venga visualizzato prima dell'immissione dell'utente. Quando si imposta la proprietà relativa al nome e/o alla password, il testo predefinito dei componenti secondari InputText per il nome e la password vengono modificati di conseguenza.

Per provare la proprietà `value` in fase di runtime, assegnare all'istanza di `LogIn` nello stage il nome **myLogin** e aggiungere il codice seguente sul fotogramma 1 nella linea temporale principale:

```
// Crea un campo di testo per la visualizzazione dei valori di login.
createTextField("myLoginValues",10,10,10,340,40)
myLoginValues.border = true;
// Gestore di eventi per l'evento click inviato per l'istanza del
// componente login.
function click(evt){
/* L'autenticazione verrebbe eseguita in questo momento.
Ad esempio il nome e password verrebbero passati a un servizio Web che
ne esegue l'autenticazione e restituisce un ID sessione e/o i ruoli di
autorizzazione attribuiti all'utente. */
myLoginValues.text = "Processing...\r";
myLoginValues.text += "Name: " + myLogin.name + " Password: " +
myLogin.password;
}

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

3. Scegliere Controllo > Prova filmato per provare il componente in Flash Player.

NOTA

Dato che il componente è in fase di prova all'interno del documento originale, è possibile che venga visualizzato un avviso relativo al fatto che è presente lo stesso identificatore di concatenamento per due simboli. Il componente funzionerà comunque. In pratica, il nuovo componente verrà utilizzato all'interno di un altro documento, nel quale caso l'identificatore di concatenamento dovrà essere univoco.

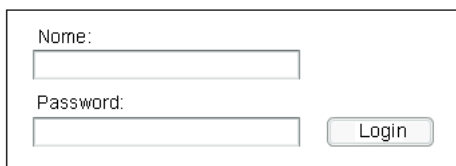
Per esportare il componente LogIn:

1. Nel file LogIn.fla, selezionare il clip filmato LogIn nella libreria e scegliere Definizione del componente dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control).
2. Selezionare Visualizza nel pannello Componenti.
3. Fare clic su OK.
4. Nel file LogIn.fla, selezionare il clip filmato LogIn nella libreria e scegliere Esporta file SWC dal menu di scelta rapida della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control).
5. Scegliere una posizione in cui salvare il file SWC.

Se il file viene salvato nella cartella di configurazione Components a livello di utente, è possibile ricaricare il pannello Componenti senza riavviare Flash e visualizzare il componente nel pannello Componenti.

NOTA

Per ulteriori informazioni sulle posizioni delle cartelle, vedere “Cartelle di configurazione installate con Flash” nella *Guida introduttiva di Flash*.



Il componente LogIn completo

Esportazione e distribuzione di un componente

I componenti vengono esportati sotto forma di pacchetti di componenti (file SWC). I componenti possono essere distribuiti come file SWC o come file FLA. Per informazioni sulla distribuzione di un componente come file FLA, vedere l'articolo su Macromedia DevNet all'indirizzo www.macromedia.com/support/flash/applications/creating_comps/creating_comps12.html.

Il modo migliore per distribuire un componente consiste nell'esportarlo in un file SWC, poiché questo tipo di file contiene tutti i file ActionScript, SWF e gli altri file necessari per utilizzare il componente. I file SWC sono utili, inoltre, se si eseguono contemporaneamente operazioni sul componente e nell'applicazione in cui viene utilizzato.

I file SWC possono essere utilizzati per la distribuzione dei componenti che vengono utilizzati in Macromedia Flash 8, Macromedia Dreamweaver MX 2004 e Macromedia Director MX 2004.

Sia che il componente venga sviluppato per uso personale o per altri utenti, è importante provare il funzionamento del file SWC nel corso dell'attività di sviluppo. Ad esempio, in un file SWC possono insorgere problemi che non si manifestano nel file FLA.

Questa sezione descrive la struttura di un file SWC e illustra come importare ed esportare i file SWC in Flash.

Nozioni fondamentali sui file SWC

Generato dallo strumento di creazione del codice di Flash, un file SWC è simile a un file .zip (compressato e poi decompresso mediante il formato di archiviazione PKZIP).

Nella tabella seguente viene descritto il contenuto di un file SWC:

File	Descrizione
catalog.xml	(Obbligatorio) Riporta il contenuto del pacchetto di componenti completo dei singoli componenti; funge da directory per gli altri file nel file SWC.
File ActionScript (AS)	Se il componente viene creato con Flash Professional 8, il codice di origine è composto da uno o più file di ActionScript contenenti una dichiarazione di classe per il componente. Quando viene esteso un componente, il compilatore utilizza il codice di origine per controllare il tipo. Il file AS non viene compilato dallo strumento di creazione, poiché il codice byte compilato si trova già nel file SWF di implementazione. Il codice di origine può contenere definizioni di classe implicite che non contengono nessun corpo di funzione, ma che vengono fornite esclusivamente per il controllo delle tipologie.
File SWF	(Obbligatori) File SWF che implementano i componenti. In un solo file SWF possono essere definiti uno o più componenti. Se il componente viene creato con Flash 8, per ogni file SWF viene esportato un solo componente.
File SWF dell'anteprim a dal vivo	(Opzionali) Se specificati, questi file SWF vengono utilizzati per l'anteprima dal vivo nello strumento di creazione del codice. Se omessi, al loro posto vengono utilizzati i file SWF che implementano il componente. I file SWF dell'anteprima dal vivo possono essere omessi quasi in tutti i casi. Dovrebbero tuttavia essere inclusi solo se l'aspetto del componente dipende da dati dinamici; ad esempio, un campo di testo che mostra il risultato di una chiamata al servizio Web.

File	Descrizione
File SWD	(Opzionale) Un file SWD corrispondente al file SWF di implementazione che consente di eseguire il debug del file SWF. Il nome del file è uguale a quello del file SWF, ma con l'estensione .swd.
File PNG	(Opzionale) Un file PNG contenente l'icona 18 x 18 pixel, 8 bit per pixel, che viene utilizzata per visualizzare un componente nelle interfacce dello strumento di creazione del codice. Se non viene fornita nessuna icona, viene usata quella predefinita. Vedere “Aggiunta di un'icona” a pagina 198 .
File SWF della finestra di ispezione Proprietà	(Opzionale) Un file SWF che viene utilizzato come finestra di ispezione Proprietà personalizzata nello strumento di creazione del codice. Se si omette il file, viene visualizzata la finestra di ispezione Proprietà predefinita.

Facoltativamente, è possibile includere anche altri file nel file SWC, dopo averlo generato dall'ambiente Flash. Ad esempio, se si desidera che gli utenti possano accedere al codice di origine del componente, si potrebbe includere un file Leggimi oppure il file FLA. Per aggiungere ulteriori file, utilizzare Macromedia Extension Manager (vedere www.macromedia.com/exchange/em_download/).

I file SWC vengono espansi in un'unica directory, pertanto, per evitare conflitti, a ogni componente deve essere assegnato un nome file univoco.

Esportazione dei file SWC

Flash offre la possibilità di esportare file SWC mediante l'esportazione di un clip filmato come file SWC. Quando si esporta un file SWC, vengono creati automaticamente dei rapporti contenenti gli errori rilevati in fase di compilazione, come se si stesse provando un'applicazione Flash.

L'esportazione di un file SWC viene effettuata per due motivi:

- Per distribuire un componente completo
- Per effettuare una prova in fase di sviluppo

Esportazione di un file SWC per un componente completo

È possibile esportare componenti come file SWC che contengono tutti i file di ActionScript, SWF e tutti gli altri file opzionali necessari per l'uso del componente.

Per esportare un file SWC per un componente completo:

1. Selezionare il clip filmato del componente nella libreria di Flash.
2. Fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) per aprire il menu di scelta rapida Libreria.
3. Selezionare Esporta file SWC dal menu di scelta rapida della libreria.
4. Salvare il file SWC.

Prova di un file SWC in fase di sviluppo

Nelle diverse fasi dell'attività di sviluppo, è consigliabile esportare il componente in un file SWC e provarne il funzionamento in un'applicazione. Se il file SWC viene esportato nella cartella Configuration/Components a livello utente, è possibile ricaricare il pannello Componenti senza chiudere e riavviare Flash.

Per provare un file SWC in fase di sviluppo:

1. Selezionare il clip filmato del componente nella libreria di Flash.
2. Fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) per aprire il menu di scelta rapida Libreria.
3. Selezionare Esporta file SWC dal menu di scelta rapida della libreria.
4. Passare alla cartella Components nella cartella Configuration a livello di utente.
Configuration/Components

NOTA

Per ulteriori informazioni sulla posizione della cartella, vedere “Cartelle di configurazione installate con Flash” nella *Guida introduttiva di Flash*.

5. Salvare il file SWC.
6. Scegliere Ricarica dal menu delle opzioni del pannello Componenti.
Il componente viene visualizzato nel pannello Componenti.
7. Trascinare il componente dal pannello Componenti in un documento.

Importazione dei file SWC dei componenti in Flash

Quando si distribuiscono i componenti ad altri sviluppatori, è possibile includere le istruzioni riportate di seguito in modo essi che possano installarli e utilizzarli immediatamente.

Per importare un file SWC:

1. Copiare il file SWC nella directory Configuration/Components.
2. Riavviare Flash.

L'icona del componente dovrebbe apparire nel pannello Componenti.

Passaggi finali dello sviluppo di un componente

Dopo aver creato il componente e averlo preparato per inserirlo nel pacchetto, è possibile aggiungervi un'icona e una descrizione. Per assicurarsi di aver completato tutti i passaggi necessari, è possibile inoltre fare riferimento a [“Elenco di controllo per lo sviluppo dei componenti” a pagina 200](#).

Aggiunta di un'icona

È possibile aggiungere un'icona che rappresenti il componente nel pannello Componenti dell'ambiente di creazione del codice di Flash.

Per aggiungere un'icona per il componente:

1. Creare una nuova immagine.

L'immagine deve misurare 18 x 18 pixel ed essere salvata in formato PNG. Deve essere a 8 bit con trasparenza alfa e il pixel in alto a sinistra deve essere trasparente per supportare l'effetto maschera.

2. Aggiungere la definizione seguente al file classe di ActionScript del componente prima della definizione della classe:

```
[IconFile("component_name.png")]
```

3. Inserire l'immagine nella stessa directory del file FLA. Quando si esporta il file SWC, Flash include l'immagine al livello principale dell'archivio.

Aggiunta di una descrizione comando

Le descrizioni comandi vengono visualizzate quando un utente passa il mouse sul nome o sull'icona di un componente nel pannello Componenti dell'ambiente di creazione del codice di Flash.

È possibile definire una descrizione comando nella finestra di dialogo Definizione del componente. Per accedere a questa finestra di dialogo, utilizzare il menu Opzioni della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control) del file FLA del componente.

Per aggiungere una descrizione comando nella finestra di dialogo Definizione del componente:

1. Aprire il file FLA del componente in Flash, assicurarsi che la libreria sia visibile (menu Finestra > Libreria).
2. Fare clic sul menu Opzioni della libreria (Windows: clic con il pulsante destro del mouse, Mac: clic tenendo premuto il tasto Control).
Il menu Opzioni della Libreria si trova a destra della barra del titolo Libreria e viene visualizzato come un'icona di tre righe con un triangolo rivolto verso il basso.
3. Scegliere l'opzione Definizione componente.
4. Nella finestra di dialogo Definizione del componente, in Opzioni, selezionare Visualizza nel pannello Componenti.
Il campo Testo descrizione comando diventa modificabile.
5. Immettere il testo della descrizione comando per il componente nella casella Testo descrizione comando.
6. Fare clic su OK per salvare le modifiche.

Elenco di controllo per lo sviluppo dei componenti

Durante la progettazione di un componente, osservare le seguenti indicazioni:

- Mantenere il più possibile ridotte le dimensioni dei file.
- Rendere il componente quanto più possibile riutilizzabile, generalizzandone la funzionalità.
- Utilizzare la classe `RectBorder` (`mx.skins.halo.RectBorder`) anziché gli elementi grafici per disegnare i bordi intorno agli oggetti. Vedere “Classe `RectBorder`” nella *Guida di riferimento dei componenti*.
- Usare l'associazione di skin basata sui tag.
- Definire le variabili `symbolName`, `symbolOwner` e `className`.
- Impostare uno stato iniziale. Poiché le proprietà di stile sono sull'oggetto, è possibile definire le impostazioni iniziali degli stili e delle proprietà, in modo che il codice di inizializzazione non debba impostarle quando viene creato l'oggetto, a meno che l'utente non sostituisca lo stato predefinito.
- Quando si definisce il simbolo, non selezionare l'opzione **Esporta** nel primo fotogramma a meno che non sia assolutamente necessario. Flash carica il componente immediatamente prima che venga utilizzato nell'applicazione Flash, pertanto, se si seleziona questa opzione, Flash precarica il componente nel primo fotogramma del principale. Solitamente è meglio evitare di precaricare il componente nel primo fotogramma per ragioni legate al Web: il componente viene caricato prima dell'avvio del precaricatore, annullando in questo modo la funzione del precaricatore.
- Evitare i clip filmato con più fotogrammi (a eccezione del livello **Assets** con due fotogrammi).
- Implementare sempre i metodi `init()` e `size()` e chiamare rispettivamente `Super.init()` e `Super.size()`; altrimenti evitare sovraccarichi.
- Evitare riferimenti assoluti, ad esempio `_root.myVariable`.
- Utilizzare `createClassObject()` invece di `attachMovie()`.
- Utilizzare `invalidate()` e `invalidateStyle()` per chiamare il metodo `draw()`, anziché chiamare esplicitamente `draw()`.
- Quando si incorporano componenti di Flash in un componente personalizzato, utilizzare i relativi simboli di filmato non compilati che si trovano nella libreria del file `StandardComponents.fla` nella cartella `Configuration/ComponentFLA`.

Proprietà della raccolta

Quando si crea un nuovo componente personalizzato in Macromedia Flash, è possibile rendere i valori delle sue proprietà disponibili per la modifica da parte dell'utente. Queste proprietà vengono definite *proprietà della raccolta*. I valori delle proprietà possono essere modificati dall'utente nella finestra di dialogo Valori (accessibile da una casella di testo all'interno della scheda Parametri del componente).

I componenti includono solitamente funzionalità per un'attività specifica, mantenendo la flessibilità necessaria per soddisfare i numerosi requisiti dell'utente del componente. Affinché un componente sia flessibile, le proprietà all'interno del componente devono a loro volta essere flessibili; in altre parole, per alcuni componenti le proprietà stesse devono poter essere modificate dall'utente del componente oltre che dai valori della proprietà.

Le proprietà della raccolta consentono di creare un numero illimitato di proprietà modificabili in un modello di oggetto. In Flash è disponibile una classe *Collection* che consente di gestire tali proprietà tramite la finestra di ispezione dei componenti.

Per l'esattezza, la classe `Collection` è una classe di tipo helper utilizzata per gestire un gruppo di oggetti correlati, ognuno dei quali viene definito *voce della raccolta*. Se si definisce una proprietà del componente come voce della raccolta, e la si rende disponibile agli utenti mediante la finestra di ispezione dei componenti, questi possono aggiungere, eliminare e modificare le voci di raccolta nella finestra di dialogo Valori durante la creazione.



Le raccolte e le voci di raccolta vengono definite come segue:

- Definire una proprietà della raccolta utilizzando il tag per metadati `Collection` nel file `ActionScript` di un componente. Per ulteriori informazioni, vedere [“Informazioni sul tag Collection” a pagina 169](#).
- Definire una voce di raccolta come classe in un file `ActionScript` separato, contenente le relative proprietà modificabili.

In Flash, le raccolte consentono di gestire più facilmente, a livello di codice, i gruppi di elementi correlati. Nelle precedenti versioni di Flash, i gruppi di elementi correlati venivano gestiti dagli autori dei componenti utilizzando più array sincronizzati a livello di codice.

Oltre alla finestra di dialogo Valori, per la gestione dei valori e delle istanze di `Collection` a livello di programmazione, in Flash sono disponibili le interfacce `Collection` e `Iterator`. Vedere [“Interfaccia Collection \(solo Flash Professional\)”](#) e [“Interfaccia Iterator \(solo Flash Professional\)”](#) nella *Guida di riferimento dei componenti*.

Questo capitolo contiene le seguenti sezioni:

Definizione della proprietà di una raccolta	203
Semplice esempio di raccolta	203
Definizione della classe per una voce della raccolta	206
Accesso alle informazioni della raccolta a livello di codice	206
Esportazione di componenti che dispongono di raccolte in un file SWC	209
Uso di un componente che dispone di una proprietà della raccolta	210

Definizione della proprietà di una raccolta

La proprietà di una raccolta viene definita utilizzando il tag per metadati Collection nel file ActionScript di un componente. Per ulteriori informazioni, vedere [“Informazioni sul tag Collection” a pagina 169](#).

NOTA

In questa sezione si presuppone che l'utente abbia familiarità con la creazione di componenti e di proprietà dei componenti modificabili.

Per definire la proprietà di una raccolta:

1. Creare un file FLA per il componente. Vedere [“Creazione del clip filmato di un componente” a pagina 149](#).
2. Creare una classe ActionScript. Vedere [“Creazione del file di classe ActionScript” a pagina 154](#).
3. Nella classe ActionScript, inserire un tag per metadati Collection. Per ulteriori informazioni, vedere [“Informazioni sul tag Collection” a pagina 169](#).
4. Definire i metodi `get` e `set` per la raccolta nel file ActionScript del componente.
5. Aggiungere le classi di utilità al file FLA selezionando Finestra > Librerie comuni > Classi, quindi trascinare `UtilsClasses` nella libreria del componente.
`UtilsClasses` include il pacchetto `mx.utils.*` per l'interfaccia `Collection`.

NOTA

Poiché `UtilsClasses` è associato al file FLA, non alla classe ActionScript, quando si controlla la sintassi durante la visualizzazione della classe ActionScript del componente, vengono generati errori del compilatore.

6. Codificare una classe contenente le proprietà delle voci di raccolta.
Vedere [“Definizione della classe per una voce della raccolta” a pagina 206](#).

Semplice esempio di raccolta

Di seguito è riportato un semplice esempio di un file di classe di un componente chiamato `MyShelf.as`. L'esempio contiene una proprietà della raccolta, oltre a un numero ridotto di importazioni, metodi e dichiarazioni per un componente che eredita dalla classe `UIObject`.

Se in questo esempio si importa `mx.utils.*`, non occorre più utilizzare i nomi di classe completi da `mx.utils`. Ad esempio, `mx.utils.Collection` può essere scritto `Collection`.

```
import mx.utils.*;
// dichiarazione di classe standard
class MyShelf extends mx.core.UIObject
{
// variabili richieste per tutte le classi
    static var symbolName:String = "MyShelf";
```

```

static var symbolOwner:Object = Object(MyShelf);
var className:String = "MyShelf";

// attributi e tag per metadati di Collection
[Collection(variable="myCompactDiscs",name="My Compact
Discs",collectionClass="mx.utils.CollectionImpl",
collectionItem="CompactDisc", identifier="Title")]

// metodi get e set per la raccolta
public function get MyCompactDiscs():mx.utils.Collection
{
    return myCompactDiscs;
}
public function set MyCompactDiscs(myCDs:mx.utils.Collection):Void
{
    myCompactDiscs = myCDs;
}

// membro privato della classe
private var myCompactDiscs:mx.utils.Collection;

// È necessario codificare un riferimento alla classe delle voci di raccolta
// per forzare l'inclusione come una dipendenza da parte del compilatore
// all'interno di SWC
private var collItem:CompactDisc;

// È necessario codificare un riferimento alla classe
mx.utils.CollectionImpl
// per forzare l'inclusione come una dipendenza da parte del compilatore
// all'interno di SWC
private var coll:mx.utils.CollectionImpl;

// metodi richiesti per tutte le classi
function init(Void):Void {
    super.init();
}
function size(Void):Void {
    super.size();
}
}

```

Per creare un file FLA di accompagnamento della classe a scopo di prova:

- 1.** In Flash, selezionare File > Nuovo e creare un documento Flash.
- 2.** Selezionare Inserisci > Nuovo simbolo. Assegnargli il nome, l'identificatore di concatenamento e il nome di classe AS 2.0 **MyShelf**.
- 3.** Deselezionare Esporta nel primo fotogramma, quindi fare clic su OK.

4. Selezionare il simbolo MyShelf nella libreria e scegliere Definizione del componente dal menu Opzioni della libreria. Immettere il nome di classe ActionScript 2.0 **MyShelf**.
5. Scegliere Finestra > Librerie comuni > Classi, quindi trascinare UtilClasses nella libreria di MyShelf.fla.
6. Nella linea temporale del simbolo MyShelf, denominare un livello **Assets**. Creare un altro livello e denominarlo **Actions**.
7. Inserire una funzione `stop()` sul fotogramma 1 del livello Actions.
8. Selezionare il fotogramma 2 del livello Assets, quindi selezionare Inserisci > Linea temporale> Fotogramma chiave.
9. Aprire il file StandardComponents.fla dalla cartella Configuration/ComponentFLA, quindi trascinare un'istanza di UIObject sullo stage di MyShelf nel fotogramma 2 del livello Assets.

Nel file FLA del componente, è necessario includere UIObject poiché, come è possibile vedere nel file di classe precedente, MyShelf estende UIObject.

10. Nel fotogramma 1 del livello Assets, disegnare un ripiano.

Può essere un semplice rettangolo; si tratta infatti semplicemente della rappresentazione visiva del componente MyShelf, che viene utilizzato a scopo di apprendimento.

11. Selezionare il clip filmato MyShelf nella libreria, quindi selezionare Converti in clip compilato.

In questo modo, è possibile trascinare il file SWF MyShelf, ovvero il clip compilato aggiunto alla libreria, nel file MyShelf.fla per provare il componente. Quando si ricompila il componente, viene visualizzata la finestra di dialogo Risolvi il conflitto tra librerie, poiché nella libreria esiste già una versione precedente del componente. Scegliere di sostituire gli elementi esistenti.

NOTA

La classe CompactDisc dovrebbe già essere stata creata; in caso contrario, quando si effettua la conversione in un clip compilato, vengono generati errori del compilatore.

Definizione della classe per una voce della raccolta

La codifica della proprietà di una voce della raccolta viene effettuata in una classe `ActionScript` separata, definita come segue:

- Definire la classe in modo che non estenda `UIObject` o `UIComponent`.
- Definire tutte le proprietà utilizzando il tag `Inspectable`.
- Definire tutte le proprietà come variabili. Non utilizzare i metodi `get` e `set` (getter/setter).

Di seguito è riportato un semplice esempio di un file di classe della voce di una raccolta, chiamato `CompactDisc.as`.

```
class CompactDisc{
    [Inspectable(type="String", defaultValue="Title")]
    var title:String;
    [Inspectable(type="String", defaultValue="Artist")]
    var artist:String;
}
```

Per visualizzare il file di classe `CompactDisc.as`, vedere [“Semplice esempio di raccolta” a pagina 203](#).

Accesso alle informazioni della raccolta a livello di codice

In Flash è possibile accedere a livello di codice ai dati della raccolta tramite le interfacce `Collection` e `Iterator`. L'interfaccia `Collection` consente di aggiungere, modificare e rimuovere voci in una raccolta. L'interfaccia `Iterator` consente di scorrere ciclicamente le voci contenute in una raccolta.

Gli scenari possibili in cui utilizzare le interfacce `Collection` e `Iterator` sono due:

- [“Accesso alle informazioni della raccolta nel file di classe \(AS\) di un componente” a pagina 207](#)
- [“Accesso alle voci di raccolta in fase di runtime in un'applicazione Flash” a pagina 208](#)

I programmatori avanzati possono, inoltre, creare, completare, accedere ed eliminare raccolte a livello di codice; per ulteriori informazioni, vedere “Interfaccia `Collection` (solo Flash Professional)” nella *Guida di riferimento dei componenti*.

Accesso alle informazioni della raccolta nel file di classe (AS) di un componente

Nel file di classe di un componente, è possibile scrivere codice che interagisca con le voci della raccolta definite durante la creazione o in fase di runtime.

Per accedere alle informazioni sulle voci di raccolta nel file di classe di un componente, è possibile utilizzare uno dei seguenti approcci:

- Il tag `Collection` include un attributo `variable`, per il quale viene specificato un tipo di variabile `mx.utils.Collection`. Utilizzare questa variabile per accedere alla raccolta, come descritto nell'esempio seguente:

```
[Collection(name="LinkButtons", variable="__linkButtons",  
    collectionClass="mx.utils.CollectionImpl", collectionItem="ButtonC",  
    identifier="ButtonLabel")]  
public var __linkButtons:mx.utils.Collection;
```

- Accedere all'interfaccia dell'iteratore per le voci di raccolta chiamando il metodo `Collection.getIterator()`, come illustrato nell'esempio seguente:

```
var itr:mx.utils.Iterator = __linkButtons.getIterator();
```

- Utilizzare l'interfaccia `Iterator` per scorrere le voci contenute nella raccolta. Il metodo `Iterator.next()` restituisce un elemento `Object`, quindi è necessario definire il tipo della voce della raccolta, come illustrato nell'esempio seguente:

```
while (itr.hasNext()) {  
    var button:ButtonC = ButtonC(itr.next());  
    ...  
}
```

- Accedere alle proprietà delle voci di raccolta, nel modo appropriato per l'applicazione in uso, come illustrato nell'esempio di seguito:

```
item.label = button.ButtonLabel;  
  
if (button.ButtonLink != undefined) {  
    item.data = button.ButtonLink;  
}  
else {  
    item.enabled = false;  
}
```

Accesso alle voci di raccolta in fase di runtime in un'applicazione Flash

Se in un'applicazione Flash viene utilizzato un componente che dispone di una proprietà della raccolta, è possibile accedere alla proprietà in fase di runtime. Con questo esempio vengono aggiunte più voci alla proprietà di una raccolta utilizzando la finestra di dialogo Valori. Le voci vengono visualizzate in runtime utilizzando le API di Collection e Iterator.

Per accedere alle voci di raccolta in fase di runtime:

1. Aprire il file MyShelf fla creato in precedenza.

Vedere [“Semplice esempio di raccolta” a pagina 203](#).

Questo esempio si basa sul componente MyShelf e sulla raccolta CompactDisc.

2. Aprire il pannello Libreria, trascinare il componente sullo stage e assegnare ad esso un nome di istanza.

In questo esempio, utilizzare il nome di istanza myShelf.

3. Selezionare il componente, aprire la finestra di ispezione dei componenti e fare clic sulla scheda Parametri. Fare clic sulla riga contenente la proprietà della raccolta, quindi fare clic sulla lente di ingrandimento a destra della riga. Viene visualizzata la finestra di dialogo Valori.
4. Immettere i valori nella proprietà della raccolta utilizzando la finestra di dialogo Valori.
5. Con il componente selezionato sullo stage, aprire il pannello Azioni e immettere il codice seguente, che deve essere associato al 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);  
    }  
}
```

Per accedere a una raccolta, utilizzare la sintassi *nomeComponente.variabileRaccolta*;
per accedere a un iteratore e scorrere le voci di raccolta, utilizzare
nomeComponente.variabileRaccolta.getIterator().

6. Selezionare Controllo > Prova filmato, quindi fare clic sullo scaffale per visualizzare i dati della raccolta nel pannello Output.

Esportazione di componenti che dispongono di raccolte in un file SWC

Quando si distribuisce un componente con una raccolta, è necessario che il file SWC contenga i seguenti file dipendenti:

- Interfaccia Collection
- Classe di implementazione di una raccolta
- Classe della voce di una raccolta
- Interfaccia Iterator

Tra questi file, di solito nel codice vengono utilizzate le interfacce Collection e Iterator, che vengono contrassegnate come classi dipendenti. I file dipendenti vengono inclusi automaticamente nel file SWC e nel file SWF di output.

Tuttavia, la classe di implementazione della raccolta (`mx.utils.CollectionImpl`) e la classe della voce della raccolta specifica del componente non vengono incluse automaticamente nel file SWC.

Per includere nel file SWC la classe di implementazione della raccolta e la classe della voce della raccolta, è necessario definire variabili private nel file `ActionScript` del componente, come descritto nell'esempio seguente:

```
// classe della voce di una raccolta
private var collItem:CompactDisc;
// classe di implementazione di una raccolta
private var coll:mx.utils.CollectionImpl;
```

Per ulteriori informazioni sui file SWC, vedere [“Nozioni fondamentali sui file SWC” a pagina 195](#).

Uso di un componente che dispone di una proprietà della raccolta

Quando si utilizza un componente nel quale è inclusa una proprietà della raccolta, per stabilire le voci contenute nella raccolta viene solitamente utilizzata la finestra di dialogo Valori.

Per utilizzare un componente che include una proprietà della raccolta:

1. Aggiungere il componente sullo stage.
2. Utilizzare la finestra di ispezione Proprietà per assegnare un nome all'istanza del componente.
3. Aprire la finestra di ispezione dei componenti e fare clic sulla scheda Parametri.
4. Fare clic sulla riga contenente la proprietà della raccolta, quindi fare clic sulla lente di ingrandimento a destra della riga.

Viene visualizzata la finestra di dialogo Valori.

5. Fare clic sul pulsante Aggiungi (+) e definire una voce della raccolta.
6. Fare clic sui pulsanti Aggiungi (+), Elimina (-) e sui pulsanti freccia per aggiungere, modificare, spostare ed eliminare le voci di raccolta.
7. Fare clic su OK.

Per ulteriori informazioni sull'accesso a una raccolta a livello di codice, vedere [“Accesso alle voci di raccolta in fase di runtime in un'applicazione Flash”](#) a pagina 208.

Indice analitico

Numeri

9 porzioni non supportato 20

A

accessibilità 21
aggiornamento dei componenti della versione 1 67
Anteprima dal vivo, funzione 64
anteprima dei componenti 64
associazione di dati con il file XML (esercitazione) 32
Associazioni, scheda dell'applicazione di esempio
(esercitazione) 33

B

broadcaster 70

C

caching bitmap non supportato 20
caricamento, componenti 66
classe Delegate (esercitazione) 79
classe DepthManager, panoramica 64
classe FocusManager, navigazione con attivazione del
componente 62
classe MovieClip, estensione 148
classe principale, selezione 145
classe WebService (esercitazione) 30
classe, fogli di stile 86
classi
creazione di riferimenti (esercitazione) 27
creazione. *Vedere* creazione dei componenti
definizione 157
ereditarietà dei componenti 18
estensione 147
importazione 157

selezione di una classe principale 145
UIComponent 147
UIObject 146
clip compilati
informazioni 20
pannello Libreria 58
clip filmato
creazione 149
definizione come componente 152
colonne
aggiunta (esercitazione) 34
colori
impostazione delle proprietà dello stile 97
personalizzazione 86
componente DataSet, associazione a XMLConnector e
DataGrid (esercitazione) 32
componente Dial (Quadrante) 135, 185
componente TextInput (esercitazione) 44
componenti 154
aggiunta ai documenti Flash 54
aggiunta di descrizioni comandi 199
aggiunta di un'icona 198
aggiunta in fase di runtime 56
anteprima 64
architettura 17
assegnazione degli skin 181
caricamento 66
categorie, descrizione 16
classe ActionScript 154
creazione di clip filmato 149
definizione del metodo draw() 177
definizione del metodo init() 174
definizione del metodo size() 177
definizione di parametri 171
definizione di variabili 159
disponibili nelle edizioni di Flash MX 12
elenco di controllo per lo sviluppo 200

- eliminazione 62
- ereditarietà 18
- esempio di creazione di un componente 135, 185
- esempio di file di classe 154
- esempio di file di classe con raccolta 203
- esportazione dei file SWC 196
- esportazione di componenti come SWC 197
- esportazione e distribuzione 194
- estensione di classi 147
- eventi 69
- eventi comuni 180
- file di origine 134
- importazione di file SWC 198
- installazione 12
- invalidazione, informazioni 178
- invio di eventi 179
- metadati, tag ComponentTask 170
- metodi getter/setter 159
- modifica di clip filmato 150
- panoramica della classe 155
- precaricamento 65
- prova di file SWC 197
- registrazione degli skin negli stili 182
- selezione di nomi di simboli 157
- selezione di una classe principale 145
- stili 182
- struttura 134
- supporto Flash Player 18
- tag per metadati
- uso in un'applicazione (esercitazione) 23
- variabile className 157
- variabile symbolOwner 157
- Vedere anche i nomi dei singoli componenti*
- componenti della versione 1, aggiornamento 67
- componenti della versione 2
 - ereditarietà 18
 - Flash Player 18
 - vantaggi 16
- Componenti, pannello 54
- components
 - creating subobjects 174
- convenzioni tipografiche 9
- createClassObject() method 174
- CSSStyleDeclaration 91, 92

D

- DataGrid, componente
 - aggiunta di colonne (esercitazione) 34
 - associazione a DataSet (esercitazione) 32
- defaultPushButton, proprietà 63
- dichiarazioni di stile
 - classe predefinita 94
 - globale 91
 - impostazione delle classi 94
 - personalizzate 92
- dispatcher (broadcaster di eventi) 70
- documentazione
 - guida alla terminologia 9
 - panoramica 8

E

- eliminazione di componenti 62
- ereditarietà nei componenti della versione 2 18
- esportazione di componenti 194
- estensione di classi 147
- Event, tag per metadati 165
- eventi
 - comuni 180
 - delega dell'area di validità 79
 - funzioni del gestore 69
 - informazioni 69
 - invio 179
 - metadati 165
 - oggetto evento 82
 - Vedere anche i nomi dei singoli componenti*

F

- file delle classi ActionScript 154
- file di classe
 - esempio 154, 203
 - informazioni 155
- file di classe del componente. *Vedere* file delle classi
- file SWC
 - clip compilati 20
 - esportazione 196
 - esportazione di proprietà della raccolta 209
 - formato di file 195
 - importazione 198
 - informazioni 20
 - prova 197
- Finestra di dialogo Valori 202
- Flash JavaScript (JSFL), tag ComponentTask 170

- Flash MX, edizioni e componenti disponibili 12
- Flash Player
 - componenti 18
 - supporto 67
- FlashType non supportato 20
- fogli di stile
 - class 86
 - personalizzate 86
- funzioni del gestore 69

G

- gestore di eventi on() 83
- griglia di dati. *Vedi* DataGrid, componente
- griglie. *Vedi* DataGrid, componente

H

- Halo, tema 115
- handleEvent, funzione di callback 74

I

- icona, per un componente 198
- identificatori di concatenamento per gli skin 102
- installazione dei componenti 12
- invio di eventi 179
- ispezione dei componenti, finestra
 - Associazioni, scheda 33
 - impostazione dei parametri 59
- istanze
 - dichiarazioni di stile 86
 - impostazione degli stili 89
- istruzione di importazione 157

L

- Label, componente
 - esercitazione 44
- libreria
 - clip compilati 58
 - Libreria, pannello 58
 - StandardComponents 150
- libreria StandardComponents 150

- listener
 - area di validità 77
 - funzioni 75
 - informazioni 70
 - oggetti 71
 - uso con i componenti (esercitazione) 35
 - uso con istanze di componenti (esercitazione)
- listener di eventi. *Vedi* listener

M

- Macromedia, materiale di riferimento aggiuntivo 9
- metadati
 - informazioni
 - tag Collection 169
 - tag ComponentTask 170
 - tag Event 165
 - tag Inspectable 162
 - tag, elenco 161
- metodi getter/setter 159
- metodo draw(), definizione 177
- metodo init(), definizione 174
- metodo invalidate() 178
- metodo size(), definizione 177

P

- pacchetti 18
- parametri dei componenti
 - definizione 171
 - impostazione 59
 - informazioni 59
 - Inspectable 162
 - visualizzazione 59
 - Vedere anche i nomi dei singoli componenti*
- parametri modificabili 162
- parametri. *Vedere* parametri dei componenti
- parola chiave 157
- parola chiave superclass 157
- personalizzazione
 - testo 86
- personalizzazione di colore e testo, utilizzando i fogli di stile 86
- precaricamento di componenti 65
- procedure ottimali per lo sviluppo dei componenti 200

- proprietà della raccolta
 - accesso a livello di codice 206
 - definizione 203
 - definizione di classi 206
 - esempio 203
 - esportazione di componenti 209
 - uso 210
- proprietà dello stile di colore 97
- Proprietà, finestra di ispezione 59
- prototype 113
- prova di file SWC 197

R

- requisiti di sistema per i componenti 8

S

- Sample, tema 115
- screen reader, accessibilità 21
- ScrollPane, componente
 - esercitazione 44
- servizio Web, connessione (esercitazione) 30
- skin
 - applicazione a componenti 108
 - applicazione ai sottocomponenti 109
 - creazione di variabili 181
 - identificatori di concatenamento
 - modifica 105
 - Vedere anche i nomi dei singoli componenti*
- skin, associazione ai componenti 101
- skin, proprietà
 - impostazione 102
 - modifica nell'oggetto prototype 113
- sottoclassi, uso per sostituire gli skin 113
- sottocomponenti, applicazione degli skin 109
- spostamento con il tasto Tab 62
- stile globale, dichiarazioni 91
- stili
 - creazione di componenti 182
 - determinazione della priorità 97
 - global, impostazione 91
 - impostazione 86
 - impostazione in un'istanza 89
 - impostazione personalizzata 92
 - informazioni 86
 - uso (esercitazione) 30
 - Vedere anche i nomi dei singoli componenti*
- subobjects, creating 174
- suggerimenti sul codice, visualizzazione 62

T

- tag Collection 169
- tag ComponentTask
 - JavaScript (JSFL) 170
- tag, *Vedere* metadati
- temi
 - applicazione 120
 - creazione 118
 - informazioni 115
- terminologia nella documentazione 9
- testo della descrizione comando, aggiunta 199
- testo, personalizzazione 86
- tipi di dati, impostazione per le istanze (esercitazione) 29

U

- UIComponent, classe
 - ereditarietà dei componenti 18
 - panoramica 147
- UIObject, classe
 - informazioni 146
- utenti finali del documento 8

V

- variabile className 157
- variabile symbolName 157
- variabile symbolOwner 157
- variabili, definizione 159
- voce della raccolta 206

X

- XMLConnector, componente
 - associazione al componente DataSet (esercitazione) 32
 - caricamento di un file XML esterno (esercitazione) 35
 - specifica dello schema (esercitazione) 32