



macromedia®

Apprendimento di ActionScript 2.0 in Flash

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

Scritto da: Jen deHaan; Peter deHaan, Joey Lott

Caporedattore: Rosana Francescato

Lead Editor: Lisa Stanziano

Redazione: Linda Adler, Geta Carlson, Evelyn Eldridge, John Hammett, Mary Kraemer, Noreen Maher, Jessie Wood, Anne Szabla

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

Un ringraziamento speciale a Jody Bleyle, Mary Burger, Lisa Friendly, Stephanie Gowin, Bonnie Loo, 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	9
A chi è destinato questo manuale.....	9
Requisiti di sistema	10
Aggiornamento dei file XML di Flash	10
Informazioni sulla documentazione	11
Altro materiale di riferimento	15
Capitolo 1: Nuove funzionalità di Flash 8 ActionScript.....	19
Novità di ActionScript 2.0 e Flash 8	19
Cambiamenti del modello di sicurezza dei file SWF installati localmente	29
Capitolo 2: Scrittura e modifica del codice ActionScript 2.0 ..	31
Informazioni su ActionScript ed eventi	32
Organizzazione del codice ActionScript	34
Uso del pannello Azioni e della finestra Script.....	36
Informazioni sul pannello Azioni	37
Informazioni sulla finestra Script.....	38
Informazioni sulla creazione di codice nel pannello Azioni e nella finestra Script	40
Informazioni sulle funzioni del pannello Azioni.....	62
Informazioni sui comportamenti	66
Informazioni sulle impostazioni di pubblicazione ActionScript	67
Capitolo 3: Informazioni su ActionScript	71
Descrizione di ActionScript	72
Informazioni sulla scelta tra ActionScript 1.0 e ActionScript 2.0	73
ActionScript e Flash Player	74
Capitolo 4: Nozioni fondamentali sul linguaggio e la sintassi ..	75
Informazioni sulla sintassi, le istruzioni e le espressioni.....	76
Informazioni sulla sintassi del punto e sui percorsi target	80
Informazioni sui segni di punteggiatura utilizzati nel linguaggio ..	88

Informazioni su costanti e parole chiave	100
Informazioni sulle istruzioni	106
Informazioni sugli array	129
Informazioni sugli operatori	143
Capitolo 5: Funzioni e metodi	169
Informazioni su funzioni e metodi	169
Nozioni fondamentali sui metodi	192
Capitolo 6: Classi	197
Informazioni sulla programmazione orientata agli oggetti e Flash ..	198
Creazione di file di classi personalizzate	208
Informazioni sulle operazioni con classi personalizzate in un'applicazione	211
Esempio: creazione di classi personalizzate	237
Esempio: uso di file di classi personalizzate in Flash.....	252
Assegnazione di una classe a simboli in Flash	255
Compilazione ed esportazione di classi	257
Nozioni fondamentali sulle classi e l'area di validità.....	260
Informazioni sulle classi incorporate e di primo livello.....	263
Informazioni sulle operazioni con le classi incorporate	273
Capitolo 7: Ereditarietà	279
Informazioni sull'ereditarietà	279
Informazioni sulla creazione di sottoclassi in Flash	281
Uso del polimorfismo in un'applicazione	287
Capitolo 8: Interfacce.....	293
Informazioni sulle interfacce	293
Creazione di interfacce come tipi di dati.....	299
Nozioni fondamentali sull'ereditarietà e le interfacce	301
Esempio: uso di interfacce	303
Esempio: creazione di un'interfaccia complessa	305
Capitolo 9: Gestione degli eventi	311
Uso dei metodi del gestore di eventi	312
Uso dei listener di eventi	314
Uso di listener di eventi con i componenti	317
Uso dei gestori di eventi pulsante e clip filmato	319
Trasmissione di eventi da istanze di componenti.....	324
Creazione di clip filmato con stati del pulsante	324

Area di validità del gestore di eventi	325
Area di validità della parola chiave this	329
Uso della classe Delegate	329
Capitolo 10: Dati e tipi di dati	333
Informazioni sui dati	333
Informazioni sui tipi di dati	334
Informazioni sulle variabili	350
Organizzazione di dati in oggetti	375
Informazioni sull'inserimento	378
Capitolo 11: Operazioni con i clip filmato	381
Informazioni sul controllo di clip filmato in ActionScript	382
Chiamata di più metodi su un singolo clip filmato	384
Caricamento e scaricamento di file SWF	385
Modifica della posizione e dell'aspetto di un clip filmato	388
Trascinamento di clip filmato	390
Creazione di clip filmato in fase di runtime	391
Aggiunta di parametri a clip filmato creati dinamicamente	396
Gestione delle profondità nei clip filmato	398
Informazioni sulla memorizzazione nella cache e sullo scorrimento dei clip filmato in ActionScript	401
Uso di clip filmato come maschere	409
Gestione di eventi clip filmato	411
Assegnazione di una classe a un simbolo clip filmato	412
Inizializzazione delle proprietà delle classi	413
Capitolo 12: Operazioni con il testo e le stringhe	415
Informazioni sui campi di testo	417
Informazioni sul caricamento di testo e variabili nei campi di testo	427
Uso di caratteri	433
Informazioni sul rendering dei caratteri e sul testo con antialiasing	443
Informazioni sul layout e sulla formattazione del testo	452
Formattazione del testo con gli stili CSS	461
Uso di un testo in formato HTML	475
Esempio: Creazione di testo scorrevole	491
Informazioni sulle stringhe e sulla classe String	492

Capitolo 13: Animazione, filtri e disegni	513
Creazione di script per animazione con ActionScript 2.0	514
Informazioni sulla memorizzazione delle bitmap nella cache, sullo scorrimento e sulle prestazioni	526
Informazioni sulle classi Tween e TransitionManager	527
Uso degli effetti filtro	545
Operazioni con i filtri mediante ActionScript	553
Gestione degli effetti filtro mediante il codice	579
Creazione di bitmap mediante la classe BitmapData	584
Informazioni sui metodi di fusione	587
Informazioni sull'ordine delle operazioni	590
Disegno con ActionScript	590
Nozioni fondamentali sulla modifica in scala e sulle guide porzione	606
 Capitolo 14: Creazione di contenuto interattivo con ActionScript	613
Informazioni su eventi e interazioni	614
Controllo della riproduzione di file SWF	614
Creazione di interattività ed effetti visivi	618
Creazione di associazioni di dati in fase di runtime mediante ActionScript	632
Analisi di uno script di esempio	641
 Capitolo 15: Operazioni con immagini, audio e video	645
Informazioni su caricamento e lavoro con contenuti multimediali esterni	646
Caricamento di file SWF e file di immagine esterni	647
Informazioni sul caricamento e l'uso di file MP3 esterni	652
Assegnazione di identificatori di concatenamento alle risorse della libreria	658
Informazioni sull'uso dei file video FLV	659
Informazioni sulla creazione di animazioni di avanzamento per file multimediali	681
 Capitolo 16: Operazioni con i dati esterni	689
Invio e caricamento di variabili	690
Uso di HTTP per la connessione a script sul lato server	695
Informazioni sul caricamento e scaricamento dei file	700
Informazioni su XML	709
Invio di messaggi verso e da Flash Player	719
Informazioni sull'API External	723

Capitolo 17: Nozioni fondamentali sulla sicurezza	733
Informazioni sulla compatibilità con i modelli di sicurezza di Flash Player precedenti	734
Informazioni sulla sicurezza dei file locali e Flash Player.....	735
Informazioni su domini, sicurezza tra domini e file SWF	753
File di criteri server-side per l'accesso ai dati.....	761
Accesso tra file SWF da HTTP a HTTPS.....	767
Capitolo 18: Esecuzione del debug delle applicazioni	771
Esecuzione del debug degli script	771
Uso del pannello Output	785
Capitolo 19: Procedure ottimali e convenzioni di codifica per ActionScript 2.0	791
Convenzioni di denominazione	793
Uso di commenti nel codice	804
Convenzioni di codifica ActionScript.....	807
Ottimizzazione di ActionScript e di Flash Player	824
Formattazione della sintassi ActionScript	825
Appendice A: Messaggi di errore	835
Appendice B: Operatori di Flash 4 obsoleti.....	841
Appendice C: Tasti della tastiera e valori dei codici tasto	843
Appendice D: Creazione di script per le versioni precedenti di Flash Player	851
Informazioni sulla sicurezza nelle versioni precedenti di Flash Player	851
Uso di Flash 8 per creare contenuto per Flash Player 4	852

Appendice E: Programmazione orientata agli oggetti con ActionScript 1.0	855
Informazioni su ActionScript 1.0	856
Creazione di un oggetto personalizzato in ActionScript 1.0	858
Assegnazione di metodi a un oggetto personalizzato in ActionScript 1.0	859
Definizione dei metodi del gestore di eventi in ActionScript 1.0	860
Creazione dell'ereditarietà in ActionScript 1.0	863
Aggiunta di proprietà getter/setter agli oggetti in ActionScript 1.0	864
Uso delle proprietà dell'oggetto Function in ActionScript 1.0	865
Appendice F: Terminologia	869
Indice analitico	877

Introduzione

Macromedia Flash Basic 8 e Macromedia Flash Professional 8 sono gli strumenti professionali standard per la creazione di codice in grado di offrire esperienze Web di grande impatto.

ActionScript è il linguaggio che consente di aggiungere interattività alle applicazioni Flash, sia che si tratti di semplici file SWF animati o di applicazioni Internet più complete e complesse. ActionScript non è necessario per utilizzare Flash; tuttavia, per creare applicazioni interattive di base o complesse, per consentire agli utenti di utilizzare altri oggetti oltre a quelli incorporati in Flash (ad esempio pulsanti e clip filmato) o per utilizzare i file SWF in modo più affidabile, l'uso di ActionScript è consigliabile.

Per ulteriori informazioni, consultare i seguenti argomenti:

A chi è destinato questo manuale	9
Aggiornamento dei file XML di Flash	10
Requisiti di sistema	10
Informazioni sulla documentazione	11
Altro materiale di riferimento	15

A chi è destinato questo manuale

In questo manuale si presume che sia già stata installata la versione base di Flash 8 o Flash Professional 8 e che l'utente abbia già una discreta familiarità con l'interfaccia utente. È opportuno conoscere le modalità di inserimento degli oggetti sullo stage e come modificarli nell'ambiente di creazione Flash. L'uso di ActionScript risulterà familiare agli utenti in grado di utilizzare un linguaggio per la creazione di script e le sue nozioni base sono di facile apprendimento anche per chi non ha dimestichezza con la programmazione. È possibile iniziare da comandi semplici e aumentare progressivamente la complessità delle operazioni. È possibile aggiungere un elevato livello di interattività ai file senza dover conoscere o scrivere molto codice.

Requisiti di sistema

ActionScript 2.0 non presenta ulteriori requisiti di sistema rispetto a Flash 8.

Nel presente manuale si suppone che per i file Flash vengano utilizzate le impostazioni di pubblicazione predefinite: Flash Player 8 e ActionScript 2.0. Se si modifica una di queste impostazioni, le procedure e gli esempi di codice illustrati nella documentazione potrebbero non funzionare correttamente. Se si sviluppano applicazioni per versioni precedenti di Flash Player, vedere [Appendice D, “Creazione di script per le versioni precedenti di Flash Player” a pagina 851](#).

Aggiornamento dei file XML di Flash

È importante che siano installati sempre i file XML di Flash più recenti. Macromedia a volte aggiunge funzionalità tramite versioni minori di Flash Player. Quando viene resa disponibile una versione di questo tipo, aggiornare la versione corrente di Flash per disporre dei file XML più aggiornati. In caso contrario, il compilatore di Flash 8 potrebbe generare errori se vengono utilizzati nuovi metodi o proprietà non disponibili nella versione di Flash Player inclusa nell'installazione di Flash.

Flash Player 7 (7.0.19.0) contiene, ad esempio, un nuovo metodo dell'oggetto System, `System.security.loadPolicyFile`. Per accedere a questo metodo, utilizzare il programma di installazione dell'aggiornamento di Flash Player per aggiornare tutte le versioni di Flash Player installate con Flash affinché il compilatore di Flash non visualizzi errori.

Tenere presente che è possibile installare un programma di aggiornamento di Flash Player più recente di una o più versioni rispetto a Flash. In questo modo, si disporrà dei file XML necessari e non verranno visualizzati errori del compilatore al momento della pubblicazione per versioni precedenti di Flash Player. A volte i nuovi metodi o le nuove proprietà possono essere utilizzati con versioni precedenti e la disponibilità dei file XML più recenti riduce al minimo gli errori del compilatore quando si tenta di accedere a metodi o proprietà meno recenti.

Informazioni sulla documentazione

Il presente manuale contiene una panoramica della sintassi di ActionScript e informazioni sull'uso di ActionScript a seconda del tipo di oggetti con cui si lavora. Per informazioni dettagliate sulla sintassi e sull'uso di ciascun elemento del linguaggio, vedere la *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “Panoramica del manuale Apprendimento di ActionScript 2.0 in Flash” a pagina 11
- “Informazioni sui file di esempio” a pagina 15
- “Termini utilizzati in questo documento” a pagina 14
- “Operazioni di copia e incolla con il codice” a pagina 14

Panoramica del manuale Apprendimento di ActionScript 2.0 in Flash

Di seguito è elencato il contenuto del manuale:

- Nel [Capitolo 1, “Nuove funzionalità di Flash 8 ActionScript”](#) sono descritte le nuove funzionalità di ActionScript, le modifiche apportate al compilatore e al debugger e il nuovo modello di programmazione del linguaggio ActionScript 2.0.
- In [Capitolo 2, “Scrittura e modifica del codice ActionScript 2.0”](#) sono descritte le funzioni dell'Editor di ActionScript, contenuto all'interno di Flash, grazie al quale la scrittura di codice risulta più semplice.
- In [Capitolo 3, “Informazioni su ActionScript”](#) viene illustrato che cos'è il linguaggio ActionScript e viene descritto in dettaglio come scegliere la versione di ActionScript da utilizzare.
- Nel [Capitolo 10, “Dati e tipi di dati”](#) sono descritti la terminologia e i concetti di base relativi a dati, tipi di dati e variabili. Questi concetti verranno utilizzati nel manuale.
- Nel [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi”](#) sono descritti la terminologia e i concetti di base del linguaggio ActionScript. Questi concetti verranno utilizzati nel manuale.
- Il [Capitolo 5, “Funzioni e metodi”](#) spiega come scrivere diversi tipi di funzioni e metodi e come utilizzarli in un'applicazione.
- Nel [Capitolo 6, “Classi”](#) viene indicato come creare classi e oggetti personalizzati in ActionScript. Nel capitolo sono inoltre elencate le classi incorporate in ActionScript e viene presentata una breve panoramica del loro utilizzo per l'accesso alle potenti funzionalità di ActionScript.

- Nel [Capitolo 7, “Ereditarietà”](#) viene descritta l'ereditarietà nel linguaggio ActionScript e vengono illustrate le procedure per estendere le classi incorporate o personalizzate.
- Nel [Capitolo 8, “Interfacce”](#) viene indicato come creare e utilizzare le interfacce in ActionScript.
- In [Capitolo 9, “Gestione degli eventi”](#) sono presentati diversi modi di gestire gli eventi: metodi del gestore di eventi, listener di eventi e gestori di eventi pulsante e clip filmato.
- In [Capitolo 11, “Operazioni con i clip filmato”](#) sono descritti i clip filmato e il codice ActionScript utilizzabile per controllarli.
- In [Capitolo 12, “Operazioni con il testo e le stringhe”](#) vengono presentati diversi modi per controllare il testo e le stringhe in Flash e informazioni sulla formattazione di testo e su FlashType (rendering del testo avanzato, come ad esempio testo con antialiasing).
- In [Capitolo 13, “Animazione, filtri e disegni”](#) vengono illustrate le procedure per creare un'animazione basata su codice e immagini, aggiungere filtri agli oggetti e disegnare mediante ActionScript.
- In [Capitolo 14, “Creazione di contenuto interattivo con ActionScript”](#) sono illustrati modi semplici per creare applicazioni con un livello di interattività avanzato, viene indicato come controllare il momento di riproduzione dei file SWF, creare puntatori personalizzati e controlli audio.
- Nel [Capitolo 15, “Operazioni con immagini, audio e video”](#) viene indicato come importare file multimediali esterni, ad esempio immagini bitmap, file MP3, file Flash Video (FLV) e altri file SWF, nelle applicazioni Flash. Questo capitolo fornisce anche una panoramica sull'utilizzo del video nelle applicazioni e su come creare una barra di avanzamento in fase di caricamento delle animazioni.
- In [Capitolo 16, “Operazioni con i dati esterni”](#) è descritta l'elaborazione di dati di origini esterne tramite script sul lato client o sul lato server inseriti nelle applicazioni. Questo capitolo illustra le procedure per l'integrazione dei dati con le applicazioni.
- Il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#) illustra la sicurezza in Flash Player, poiché attiene all'utilizzo dei file SWF localmente sul disco rigido. Vengono inoltre illustrati i problemi di sicurezza tra domini e viene spiegato come caricare dati dai server, o tra domini.
- Nel [Capitolo 18, “Esecuzione del debug delle applicazioni”](#) è descritto il debugger di ActionScript, contenuto all'interno di Flash, grazie al quale la scrittura delle applicazioni risulta più semplice.
- In [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0”](#) sono contenute le procedure ottimali per l'utilizzo di Flash e la scrittura di codice ActionScript. Questo capitolo elenca inoltre le convenzioni di codifica standard, come ad esempio assegnazione di nomi alle variabili ed altre convenzioni.

- In [Appendice A, “Messaggi di errore”](#) sono elencati i messaggi di errore che possono essere generati dal compilatore Flash.
- In [Appendice B, “Operatori di Flash 4 obsoleti”](#) sono elencati tutti gli operatori di Flash 4 obsoleti e le relative modalità di associazione.
- Nell'[Appendice C, “Tasti della tastiera e valori dei codici tasto”](#) è contenuto l'elenco di tutti i tasti di una tastiera standard e i corrispondenti valori ASCII da utilizzare per identificare i tasti in ActionScript.
- Nell'[Appendice D, “Creazione di script per le versioni precedenti di Flash Player”](#) sono fornite linee guida per facilitare la scrittura di script con sintassi corretta per la versione di Flash Player che si desidera impostare come destinazione.
- Nell'[Appendice E, “Programmazione orientata agli oggetti con ActionScript 1.0”](#) vengono presentate informazioni sull'uso del modello a oggetti di ActionScript 1.0 per la scrittura di script.
- Nell'[Appendice F, “Terminologia”](#) viene elencata la terminologia utilizzata comunemente quando si usa il linguaggio ActionScript e viene fornita la descrizione dei relativi termini.

In questo manuale viene illustrato l'uso del linguaggio ActionScript. Per informazioni sugli elementi del linguaggio, vedere la *Guida di riferimento di ActionScript 2.0*.

Convenzioni tipografiche

Questo manuale utilizza le seguenti convenzioni tipografiche:

- Carattere codice indica una stringa di codice ActionScript.
- Carattere codice grassetto Viene utilizzato in genere all'interno di una procedura e indica codice che deve essere modificato o aggiunto ad altro codice già presente nel file FLA. In alcuni casi potrebbe essere utilizzato per evidenziare codice che necessita di attenzione.
- Testo in grassetto Indica dati da immettere nell'interfaccia utente, ad esempio un nome di file o un nome di istanza.
- *Testo in corsivo* Indica un nuovo termine definito nel testo seguente. In un percorso di un file potrebbe indicare un valore da sostituire, ad esempio con il nome di una directory presente sul disco rigido.

Termini utilizzati in questo documento

In questo manuale vengono utilizzati i termini seguenti:

- *Sviluppatore* È l'autore degli script o delle applicazioni.
- *Utente* È la persona che esegue gli script e le applicazioni creati dallo sviluppatore.
- *Fase di compilazione* È la fase nella quale il documento viene pubblicato, esportato, verificato e sottoposto a debug.
- *Runtime* È la fase nella quale lo script viene eseguito in Flash Player.

I termini specifici di ActionScript come *metodo* e *oggetto* sono definiti nell'[Appendice F, “Terminologia” a pagina 869](#).

Operazioni di copia e incolla con il codice

Quando si incolla ActionScript dal pannello della Guida nel file FLA o ActionScript, è necessario prestare attenzione ai caratteri speciali. I caratteri speciali includono virgolette speciali (dette anche virgolette inglesi o virgolette intelligenti). Questi caratteri non vengono interpretati dall'Editor di ActionScript e quindi il codice genera un errore se si cerca di compilarlo in Flash.

È possibile stabilire che i caratteri virgolette siano caratteri speciali se non hanno il colore del codice corretto. Ovvero, se tutte le stringhe non cambiano di colore nell'editor di codice, è necessario sostituire i caratteri speciali con caratteri virgolette diritte normali. Se si digita un carattere virgoletta doppia o virgoletta semplice direttamente nell'Editor di ActionScript, viene digitato sempre un carattere virgoletta diritta. Il compilatore (quando si prova o si pubblica un file SWF) genera un errore e consente di capire se nel codice esista il tipo errato (virgolette speciali o virgolette inglesi) di caratteri.



È inoltre possibile incontrare virgolette speciali se si incolla ActionScript da altre posizioni, come ad esempio una pagina Web o un documento Microsoft Word.

Prestare molta attenzione alle interruzioni di riga appropriate nelle operazioni di copia e incolla con il codice. Se si incolla il codice da posizioni diverse, la riga di codice potrebbe interrompersi in una posizione scorretta. Assicurarsi che la codifica mediante colori della sintassi sia corretta nell'Editor di ActionScript, se si ha motivo di ritenere che le interruzioni di riga possano rappresentare un problema. È possibile confrontare il codice nel pannello Azioni con quello nel pannello della Guida per verificarne la corrispondenza. Abilitare A capo automatico nell'Editor di ActionScript per risolvere l'eccesso di interruzioni di riga nel codice (selezionare Visualizza > A capo automatico nella finestra Script, o A capo automatico dal menu a comparsa del pannello Azioni).

Altro materiale di riferimento

Oltre al presente manuale su ActionScript, sono disponibili manuali su altri argomenti relativi a Flash, come ad esempio i componenti e Macromedia Flash Lite. È possibile accedere a ciascun manuale nel pannello della Guida (? > Guida di Flash), visualizzando l'indice predefinito. Fare clic sul pulsante Cancella per visualizzare tutti i manuali disponibili; per ulteriori informazioni, vedere “[Dove trovare documentazione su altri argomenti](#)” a pagina 18.

Per ulteriori informazioni sulle altre risorse disponibili, consultare i seguenti argomenti:

- “[Informazioni sui file di esempio](#)” a pagina 15
- “[Dove trovare i file PDF o la documentazione stampata](#)” a pagina 16
- “[Informazioni su LiveDocs](#)” a pagina 16
- “[Ulteriori risorse online](#)” a pagina 18
- “[Dove trovare documentazione su altri argomenti](#)” a pagina 18

Informazioni sui file di esempio

Esistono numerosi file di esempio basati su ActionScript che vengono installati con Flash. I suddetti file di esempio illustrano il funzionamento del codice in un file FLA; spesso questo si rivela un utile strumento di apprendimento. I capitoli di questo manuale spesso fanno riferimento a questi file, ma si consiglia di esaminare comunque la cartella dei file di esempio sul disco rigido.

I file di esempio includono file di applicazioni FLA che utilizzano funzionalità Flash comuni installati con Flash. Queste applicazioni sono state concepite per presentare le caratteristiche delle applicazioni Flash agli sviluppatori che ancora non hanno familiarità con il programma e illustrare agli sviluppatori più esperti le funzionalità di Flash all'interno di un contesto.

È possibile trovare i file sorgente di esempio incentrati su ActionScript nella cartella Samples sul disco rigido.

- In Windows, accedere a *Unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/*.

I seguenti file di esempio incentrati sui componenti potrebbero rivelarsi utili, poiché contengono molto codice ActionScript. Si trovano inoltre nella cartella Samples sul disco rigido:

- In Windows, accedere a *Unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/*.

È possibile anche scaricare da Internet ulteriori file di esempio. La seguente pagina Web contiene collegamenti e descrizioni di ulteriori file di esempio: www.macromedia.com/go/flash_samples/.

Dove trovare i file PDF o la documentazione stampata

Se si preferisce leggere la documentazione in formato stampato, sono disponibili per lo scaricamento le versioni PDF di ciascun manuale della Guida. Visitare il sito Web www.macromedia.com/support/documentation_it/ e selezionare il prodotto che interessa. È possibile visualizzare o scaricare la versione PDF o creare un collegamento con la versione LiveDocs del manuale.

Spesso è possibile anche acquistare la documentazione stampata. Per le informazioni aggiornate, consultare il [Sito di supporto della documentazione](#) e selezionare Flash Basic 8 o Flash Professional 8.

Informazioni su LiveDocs

È possibile accedere alla documentazione nel sito Web LiveDocs oltre che accedere ad essa dal pannello della Guida. Il sito Web LiveDocs contiene tutte le pagine della Guida di Flash e potrebbe contenere commenti che chiariscono, aggiornano, o correggono parti della documentazione. Fare clic su View Comments on LiveDocs nella parte inferiore di una pagina nel pannello della Guida per visualizzare la pagina del sito Web LiveDocs equivalente. Visitare il sito Web <http://livedocs.macromedia.com> per visualizzare un elenco della documentazione disponibile nel formato LiveDocs.

Gli autori della documentazione tecnica sono abituali frequentatori del sito Web. Uno dei vantaggi di LiveDocs consiste nel vedere dei commenti che chiariscono il contenuto della documentazione, oppure correggono eventuali errori o problemi che insorgono nell'ambito di una particolare release del software. LiveDocs non è il luogo giusto per sottoporre richieste di assistenza, ad esempio riportando esempi di codice che non funziona, commenti a problemi relativi al software o all'installazione, o ponendo domande su come creare qualcosa con Flash. È invece il luogo appropriato per inviare commenti sulla documentazione (ad esempio, se si nota una frase o un paragrafo che potrebbe essere formulato in maniera più chiara).

Quando si fa clic sul pulsante per aggiungere un commento in LiveDocs, sono disponibili varie indicazioni sui tipi di commenti considerati accettabili nel sistema. Leggere attentamente queste informazioni, altrimenti il commento inserito potrebbe essere rimosso dal sito Web.

Per inviare domande relative all'applicazione, utilizzare i forum Web di Macromedia:
www.macromedia.com/support/forum_it/. I forum sono il luogo perfetto per sottoporre domande, perché molti membri del personale Macromedia, volontari del Team Macromedia, gestori e membri dei gruppi di utenti Macromedia e persino autori della documentazione tecnica sono abituali frequentatori dei forum.

I tecnici non si occupano del monitoraggio del sistema LiveDocs, bensì della Wish List di Flash. Se si ritiene di avere scoperto un errore nel software, o si desidera richiedere un miglioramento di Flash, compilare l'apposito modulo all'indirizzo Web
www.macromedia.com/go/wish. La semplice segnalazione di un errore o di una richiesta di miglioramento su LiveDocs, non basta per inserire l'errore o la richiesta nel database degli errori. Affinché un tecnico prenda in esame l'errore o la richiesta segnalata, è necessario utilizzare il modulo appositamente predisposto.

Prestare attenzione ai caratteri speciali e alle interruzioni di riga quando si incolla dal Web, incluso LiveDocs. Macromedia ha fatto il possibile per eliminare tutti i caratteri speciali dagli esempi di codice, ma in caso di problemi con le operazioni di incolla con il codice, consultare “Operazioni di copia e incolla con il codice” a pagina 14.

Ulteriori risorse online

Svariate risorse online offrono un'ampia varietà di istruzioni, informazioni e consigli per l'utilizzo di Macromedia Flash 8. Si consiglia di visitare spesso i siti Web seguenti per verificare la presenza di aggiornamenti:

Il sito Web Centro per sviluppatori Macromedia (www.macromedia.com/devnet) viene aggiornato periodicamente con le informazioni più recenti su Flash e riporta consigli di utenti esperti, argomenti avanzati, esempi, suggerimenti, esercitazioni anche suddivise in diverse parti e aggiornamenti vari. Per reperire le ultime notizie su Flash e usare al meglio il programma, visitare periodicamente il sito.

Nel **Centro di assistenza Macromedia Flash** (www.macromedia.com/support/flash) sono disponibili note tecniche, aggiornamenti della documentazione e collegamenti ad altre risorse nella comunità degli utenti Flash.

Il sito Web Macromedia Weblogs (<http://weblogs.macromedia.com>) comprende un elenco di weblog (detti anche *blog*) inseriti sia da dipendenti Macromedia, sia da utenti della comunità.

Il sito Macromedia Online Forums (<http://webforums.macromedia.com>) contiene numerosi forum in cui vengono poste domande specifiche su Flash, le applicazioni o il linguaggio ActionScript. I forum sono moderati da volontari del team Macromedia e spesso vi partecipano anche dipendenti Macromedia. Per dubbi sulle informazioni da consultare o se non si conosce la soluzione a un problema, un forum di Flash può essere un buon punto di partenza.

Il sito Web Macromedia Community (www.macromedia.com/community) ospita regolarmente Macrochat, una serie di presentazioni dal vivo su un'ampia varietà di argomenti eseguite da dipendenti Macromedia o membri della comunità. Si consiglia di visitare spesso questo sito Web per verificare la presenza di aggiornamenti e di registrarsi per poter assistere alle macrochat.

Dove trovare documentazione su altri argomenti

I seguenti manuali offrono informazioni supplementari su argomenti comunemente associati ad ActionScript 2.0:

- Per informazioni sugli elementi che compongono il linguaggio ActionScript, vedere la *Guida di riferimento di ActionScript 2.0*.
- Per informazioni sull'uso dell'ambiente di creazione di Flash, consultare la *Guida all'uso*.
- Per informazioni sull'uso dei componenti, consultare *Uso dei componenti*.

Nuove funzionalità di Flash 8 ActionScript

Macromedia Flash Basic 8 e Macromedia Flash Professional 8 offrono nuove funzioni che rendono estremamente semplice la scrittura di script affidabili in linguaggio ActionScript (AS). Le nuove funzioni, descritte in questo capitolo, comprendono nuovi elementi di linguaggio (vedere “[Nuove funzioni aggiunte al linguaggio ActionScript](#)” a pagina 22), strumenti di editing migliorati (vedere “[Modifiche all'Editor di ActionScript](#)” a pagina 28), modifiche al modello di sicurezza e nuove funzioni ActionScript per lo strumento di creazione.

Per ulteriori informazioni, consultare i seguenti argomenti:

Novità di ActionScript 2.0 e Flash 8	19
Cambiamenti del modello di sicurezza dei file SWF installati localmente	29

Novità di ActionScript 2.0 e Flash 8

Il linguaggio ActionScript è stato ulteriormente sviluppato a partire dalla sua introduzione avvenuta diversi anni fa. In ogni nuova versione di Flash sono stati aggiunti al linguaggio ActionScript nuovi metodi, parole chiave, oggetti e altri elementi. Sono disponibili anche miglioramenti relativi ad ActionScript per gli ambienti di creazione Flash 8. Flash Basic 8 e Flash Professional 8 introducono una serie di nuovi elementi di linguaggio che consentono di applicare funzioni come filtri e metodi di fusione, oltre a funzioni studiate per lo sviluppo di applicazioni, quali l'integrazione JavaScript (ExternalInterface) e l'I/O file (FileReference e FileReferenceList).

Questa sezione fornisce una panoramica degli elementi e delle classi del linguaggio ActionScript nuovi o modificati in Flash 8 e miglioramenti relativi ad ActionScript per lo strumento di creazione. Per un elenco delle nuove funzioni specifiche per ActionScript 2.0, consultare “[Nuove funzioni aggiunte al linguaggio ActionScript](#)” a pagina 22. Per utilizzare negli script uno qualsiasi dei nuovi elementi del linguaggio è necessario utilizzare Flash Player 8 (l'impostazione predefinita) quando si pubblicano i documenti.

Le seguenti funzioni sono state aggiunte sia a Flash Basic 8 che a Flash Professional 8 (a meno che non venga indicato diversamente):

- I miglioramenti all'editor di ActionScript consentono di mostrare negli script i caratteri nascosti. Per ulteriori informazioni, vedere “[Visualizzazione dei caratteri nascosti](#)” a pagina 58.
- Ora le opzioni di debug per i file di ActionScript sono disponibili sia nella finestra Script che nel pannello Azioni.
- La directory Configuration che comprende i file XML e Class è stata riorganizzata. Per ulteriori informazioni, vedere “[File di configurazione installati con Flash 8](#)” a pagina 70.
- Quando si lavora su un'applicazione è possibile impostare una preferenza per ricaricare i file di script modificati, operazione che impedisce di lavorare con versioni datate dei file di script e di sovrascrivere quelli più recenti. Per ulteriori informazioni, vedere “[Informazioni sulle preferenze di ActionScript](#)” a pagina 45.
- La funzione della finestra Script è disponibile sia in Flash Basic 8 che in Flash Professional 8. Questo significa che ora è possibile creare un file ActionScript con entrambi i programmi.
- Assistente script (simile alla modalità Normale nelle versioni precedenti di Flash) aiuta a scrivere il codice senza necessariamente comprendere la sintassi. Per ulteriori informazioni su Assistente script, vedere “[Informazioni su Assistente script](#)” a pagina 63.
- In fase di runtime è possibile caricare nuovi tipi di file di immagine, tra cui immagini JPEG progressive, GIF non animate e file PNG. Se si carica un file animato apparirà il primo fotogramma dell'animazione.
- È possibile assegnare identificatori di concatenamento ai file bitmap e audio memorizzati nella Libreria, cioè si possono associare immagini allo stage o lavorare con queste risorse nelle librerie condivise.
- La memorizzazione delle bitmap consente di migliorare le prestazioni delle applicazioni in fase di runtime memorizzando una rappresentazione bitmap delle istanze. È possibile utilizzare il codice ActionScript per accedere a questa proprietà. Per ulteriori informazioni, vedere “[Informazioni sulla memorizzazione delle bitmap nella cache, sullo scorrimento e sulle prestazioni](#)” a pagina 526.

- La modifica in scala a 9 porzioni consente di modificare in scala le istanze di clip filmato senza ampliare i tratti che delineano il clip filmato. È possibile utilizzare il codice ActionScript per accedere a questa funzione sia in Flash Basic 8 che in Flash Professional 8, o nello strumento di creazione Flash 8. Per ulteriori informazioni, vedere “[Operazioni con la modifica in scala a 9 porzioni in ActionScript](#)” a pagina 609. Per ulteriori informazioni su come accedere alla modifica in scala a 9 porzioni nello strumento di creazione vedere “[Informazioni sulla modifica in scala a 9 porzioni e sui simboli di clip filmato](#)” a pagina 92 nella guida *Uso di Flash*.
- Nella finestra di dialogo Impostazioni pubblicazione ora è possibile aggiungere ai file FLA le informazioni sui metadati. È possibile inserire nei file FLA un nome e una descrizione utilizzando la finestra di dialogo per aumentare la visibilità della ricerca online.
- Il pannello Stringhe è stato migliorato così da includere il supporto multiriga nel campo Stringa e nel file del linguaggio XML. Per ulteriori informazioni, vedere “[Informazioni sul pannello Stringhe](#)” a pagina 494.
- In Flash Player è stato incorporato un nuovo garbage collector, che per migliorare le prestazioni utilizza un raccoglitrice incrementale.
- Il flusso di lavoro per la creazione di applicazioni accessibili è stato migliorato. Flash Player 8 non necessita più degli sviluppatori per aggiungere tutti gli oggetti all'indice di tabulazione per il contenuto che deve essere letto correttamente da uno screen reader. Per ulteriori informazioni sull'indice di tabulazione vedere tabIndex (proprietà Button.tabIndex), tabIndex (proprietà MovieClip.tabIndex) e tabIndex (proprietà TextField.tabIndex) nella *Guida di riferimento di ActionScript 2.0*.
- Flash Player ha migliorato la sicurezza per i file locali, per una maggiore sicurezza durante l'esecuzione dei file SWF sul disco rigido. Per informazioni sulla sicurezza dei file locali, vedere “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.
- Utilizzando il codice ActionScript è possibile sfruttare l'API di disegno per controllare lo stile di linea dei tratti che vengono disegnati. Per informazioni sui nuovi stili di linea, vedere “[Uso degli stili di linea](#)” a pagina 598.
- Utilizzando il codice ActionScript è possibile sfruttare l'API di disegno per creare gradienti più complessi con cui riempire le forme. Per ulteriori informazioni sui riempimenti con gradiente, vedere “[Uso di riempimenti con gradiente complessi](#)” a pagina 597.
- È possibile utilizzare il codice ActionScript per applicare diversi filtri agli oggetti sullo stage (come le istanze di clip filmato). Per informazioni sui filtri e su ActionScript, vedere “[Operazioni con i filtri mediante ActionScript](#)” a pagina 553.
- È possibile utilizzare l'API FileReference e FileReferenceList per caricare i file su un server. Per ulteriori informazioni, vedere “[Informazioni sul caricamento e scaricamento dei file](#)” a pagina 700.

- È possibile utilizzare il codice ActionScript per accedere a metodi nuovi e avanzati per l'applicazione e la manipolazione dei colori. Per ulteriori informazioni, vedere [“Impostazione dei valori dei colori” a pagina 624](#) e `ColorTransform` (`flash.geom.ColorTransform`) nella *Guida di riferimento di ActionScript 2.0*.
- Sono stati eseguiti diversi miglioramenti alla gestione del testo, tra cui nuove opzioni, proprietà e parametri nelle classi `TextField` e `TextFormat`. Per ulteriori informazioni, vedere `TextField` e `TextFormat` nella *Guida di riferimento di ActionScript 2.0*.
- È possibile utilizzare il codice ActionScript per accedere alle funzioni avanzate di antialiasing (`FlashType`). Per ulteriori informazioni, vedere [“Informazioni sul rendering dei caratteri e sul testo con antialiasing” a pagina 443](#).
- Quando si verifica l'applicazione è possibile cancellare i file ASO. Premere Controllo > Elimina file ASO o Controllo > Elimina file ASO e Prova filmato nello strumento di creazione. Per ulteriori informazioni, vedere [“Utilizzo dei file ASO” a pagina 258](#).

Per un elenco di classi, elementi di linguaggio, metodi e proprietà specifiche aggiunte ad ActionScript 2.0 in Flash 8, vedere [“Nuove funzioni aggiunte al linguaggio ActionScript” a pagina 22](#).

Nuove funzioni aggiunte al linguaggio ActionScript

Questa sezione elenca le aggiunte eseguite agli elementi del linguaggio e alle classi ActionScript nuove o che sono state modificate in Flash 8. Le classi e gli elementi di linguaggio seguenti sono nuove funzioni oppure non erano mai state supportate in precedenza da Flash Player 8.

Le seguenti classi sono state aggiunte ad ActionScript 2.0 in Flash 8:

- La classe `BevelFilter` (nel pacchetto `flash.filters`) consente di aggiungere agli oggetti effetti di smussatura.
- La classe `BitmapData` (nel pacchetto `flash.display`) consente di creare e manipolare immagini bitmap trasparenti od opache di dimensioni arbitrarie.
- La classe `BitmapFilter` (nel pacchetto `flash.display`) è una classe base per gli effetti filtro.
- La classe `BlurFilter` consente di applicare le sfocature agli oggetti in Flash.
- La classe `ColorMatrixFilter` (nel pacchetto `flash.filters`) consente di applicare le trasformazioni ai colori ARGB e ai valori alfa.
- La classe `ColorTransform` (nel pacchetto `flash.geom`) consente di regolare i valori di colore nei clip filmato. La classe `Color` è sconsigliata; al suo posto viene utilizzata questa classe.
- La classe `ConvolutionFilter` (nel package `flash.filters`) consente di applicare effetti filtro di convoluzione matrice.

- La classe DisplacementMapFilter (nel pacchetto flash.filters) consente di utilizzare i valori in pixel di un oggetto BitmapData per eseguire lo spostamento su un oggetto.
- La classe DropShadowFilter (nel pacchetto flash.filters) consente di aggiungere agli oggetti ombre esterne.
- La classe ExternalInterface (nel pacchetto flash.external) consente di comunicare utilizzando ActionScript con il contenitore Flash Player (il sistema contenente l'applicazione Flash, come un browser con JavaScript o l'applicazione desktop).
- La classe FileReference (nel pacchetto flash.net) consente di caricare e scaricare i file tra il computer dell'utente e un server.
- La classe FileReferenceList (nel pacchetto flash.net) consente di selezionare uno o più file da caricare.
- La classe GlowFilter (nel pacchetto flash.filters) consente di aggiungere agli oggetti effetti di bagliore.
- La classe GradientBevelFilter (nel pacchetto flash.filters) consente di aggiungere agli oggetti smussature con gradiente.
- La classe GradientGlowFilter (nel pacchetto flash.filters) consente di aggiungere agli oggetti effetti di bagliore con gradiente.
- La classe IME (nella classe System) consente di manipolare l'IME (Input Method Editor) del sistema operativo all'interno di Flash Player.
- La classe Locale (nel pacchetto mx.lang) consente di controllare come appare il testo in più lingue in un file SWF.
- La classe Matrix (nel pacchetto flash.geom) rappresenta una matrice di trasformazione che determina come mappare punti da uno spazio di coordinate a un altro.
- La classe Point (nel pacchetto flash.geom) rappresenta una posizione in un sistema di coordinate a due dimensioni (*x* rappresenta l'asse orizzontale e *y* rappresenta l'asse verticale).
- La classe Rectangle (nel pacchetto flash.geom) consente di creare e modificare gli oggetti Rectangle.
- La classe TextRenderer (nel pacchetto flash.text) fornisce la funzionalità per l'antialiasing dei caratteri incorporati.
- La classe Transform (nel pacchetto flash.geom) raccoglie i dati sulle trasformazioni di colore e le modifiche delle coordinate applicate a un'istanza MovieClip.

NOTA

In Flash 8 è stato aggiunto il supporto ufficiale alla classe AsBroadcaster.

I nuovi elementi di linguaggio, metodi e funzioni aggiunti alle classi esistenti in ActionScript comprendono:

- La funzione globale `showRedrawRegions` offre al player di debugger la possibilità di evidenziare le aree dello schermo che vengono ridisegnate (cioè, aree poco pulite che vengono aggiornate). La funzione permette al player di mostrare ciò che è stato ridisegnato, ma non consente di controllare tali aree.
- La proprietà `blendMode` della classe `Button`, che imposta il metodo di fusione per l'istanza del pulsante.
- La proprietà `cacheAsBitmap` della classe `Button`, che consente di memorizzare l'oggetto nella cache come rappresentazione bitmap interna dell'istanza.
- La proprietà `filters` della classe `Button`, che rappresenta un array indicizzato contenente ciascun oggetto filtro associato al pulsante.
- La proprietà `scale9Grid` della classe `Button`, che rappresenta l'area rettangolare che definisce le nove aree di modifica in scala per l'istanza.
- La proprietà `hasIME` della classe `System.capabilities`, che indica se nel sistema è stato installato un IME.
- La proprietà `getUTCYear` della classe `Date`, che restituisce l'anno corrente in base all'ora universale.
- Il metodo `isAccessible()` della classe `Key` restituisce un valore booleano che indica se l'ultimo tasto premuto è accessibile da altri file SWF, in base alle restrizioni di sicurezza.
- Il gestore di eventi `onHTTPStatus` della classe `LoadVars` restituisce il codice di stato inviato dal server (per esempio, il valore 404 per una pagina non trovata). Per ulteriori informazioni, vedere `onHTTPStatus` (gestore `LoadVars.onHTTPStatus`) nella *Guida di riferimento di ActionScript 2.0*.
- Il metodo `attachBitmap()` della classe `MovieClip`, che associa un'immagine bitmap a un clip filmato. Per ulteriori informazioni, vedere `BitmapData` (`flash.display.BitmapData`) nella *Guida di riferimento di ActionScript 2.0*.
- Il metodo `beginBitmapFill()` della classe `MovieClip`, che riempie un clip filmato con un'immagine bitmap.
- I parametri `spreadMethod`, `interpolationMethod` e `focalPointRatio` del metodo `beginGradientFill()` della classe `MovieClip`. Questo metodo riempie un'area di disegno con un'immagine bitmap, e la bitmap può essere ripetuta o affiancata a mosaico per riempire l'area.
- La proprietà `blendMode` della classe `MovieClip`, che consente di impostare il metodo di fusione per l'istanza.

- La proprietà `cacheAsBitmap` della classe `MovieClip`, che consente di memorizzare l'oggetto nella cache come rappresentazione bitmap interna dell'istanza.
- La proprietà `filters` della classe `MovieClip`, che rappresenta un array indicizzato contenente ciascun oggetto filtro correntemente associato all'istanza.
- Il metodo `getRect()` della classe `MovieClip`, che restituisce le proprietà che rappresentano i valori minimi e massimi delle coordinate dell'istanza specificata.
- Il metodo `lineGradientStyle()` della classe `MovieClip`, che specifica uno stile di linea sfumato utilizzato da Flash quando disegna un percorso.
- I parametri `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` e `miterLimit` del metodo `lineStyle()` della classe `MovieClip`. Questi parametri specificano i tipi di stile di linea utilizzabili quando si disegnano le linee.
- La proprietà `opaqueBackground` della classe `MovieClip`, che imposta il colore dello sfondo opaco (non trasparente) del clip filmato sul colore specificato dal valore RGB esadecimale.
- La proprietà `scale9Grid` della classe `MovieClip`, che rappresenta l'area rettangolare che definisce le nove aree di modifica in scala per l'istanza.
- La proprietà `scrollRect` della classe `MovieClip`, che consente di scorrere velocemente il contenuto del clip filmato per ottenere una visualizzazione più estesa del contenuto nella finestra.
- La proprietà `transform` della classe `MovieClip`, che consente di eseguire le impostazioni riguardanti la matrice, la trasformazione del colore e i limiti di pixel di un clip filmato. Per ulteriori informazioni, vedere `Transform` (`flash.geom.Transform`) nella *Guida di riferimento di ActionScript 2.0*.
- Il parametro `status` di `MovieClipLoader`. Il gestore di eventi `onLoadComplete` restituisce il codice di stato inviato dal server (per esempio, il valore `404` per una pagina non trovata). Per ulteriori informazioni, vedere `onLoadComplete` (listener di eventi `MovieClipLoader.onLoadComplete`) nella *Guida di riferimento di ActionScript 2.0*.
- Il gestore di eventi `onLoadError` della classe `MovieClipLoader` viene chiamato quando l'operazione di caricamento di un file con `MovieClipLoader.loadClip()` fallisce.
- Il parametro `secure` del metodo `SharedObject.getLocal()` stabilisce se l'accesso a questo oggetto condiviso è limitato ai file SWF inviati attraverso una connessione HTTPS. Per ulteriori informazioni, vedere `getLocal` (metodo `SharedObject.getLocal`) nella *Guida di riferimento di ActionScript 2.0*.
- La proprietà `sandboxType` della classe `System.security` indica il tipo di funzione di sicurezza sandbox in cui opera il file SWF chiamante. Per ulteriori informazioni, vedere `sandboxType` (proprietà `security.sandboxType`) nella *Guida di riferimento di ActionScript 2.0*.

- La proprietà `antiAliasType` della classe `TextField`, che imposta il tipo di antialiasing utilizzato per l'istanza `TextField`.
- La proprietà `filters` della classe `TextField`, che rappresenta un array indicizzato contenente ciascun oggetto filtro correntemente associato all'istanza `TextField`.
- La proprietà `gridFitType` della classe `TextField`, che imposta il tipo di adattamento alla griglia utilizzato per l'istanza. Per ulteriori informazioni sull'adattamento alla griglia e su `TextField.gridFitType`, vedere `gridFitType` (proprietà `TextField.gridFitType`) nella *Guida di riferimento di ActionScript 2.0*.
- La proprietà `sharpness` della classe `TextField`, che imposta la precisione dei bordi del glifo per l'istanza `TextField`. Se si utilizza questa proprietà è necessario impostare il metodo `antiAliasType()` su `advanced`.
- La proprietà `thickness` della classe `TextField`, che imposta lo spessore dei bordi del glifo per l'istanza `TextField`. Se si utilizza questa proprietà è necessario impostare il metodo `antiAliasType()` su `advanced`.
- Il valore `justify` per la proprietà `align` della classe `TextFormat`, che consente di giustificare un paragrafo specifico.
- La proprietà `indent` della classe `TextFormat`, che consente di utilizzare valori negativi.
- La proprietà `kerning` della classe `TextFormat`, che consente di attivare o disattivare la crenatura per l'oggetto `TextFormat`.
- La proprietà `leading` della classe `TextFormat`, che consente di utilizzare un'interlinea negativa così che lo spazio tra le righe sia inferiore all'altezza del testo. In questo modo nelle applicazioni è possibile avvicinare le righe del testo.
- La proprietà `letterSpacing` della classe `TextFormat`, che consente di specificare la quantità di spazio uniformemente distribuita tra i caratteri.
- La proprietà `_alpha` della classe `Video`, che rappresenta la quantità specifica di trasparenza per l'oggetto video.
- La proprietà `_height` della classe `Video`, che indica l'altezza dell'istanza video.
- La proprietà `_name` della classe `Video`, che indica il nome di istanza del video.
- La proprietà `_parent` della classe `Video`, che indica l'istanza o l'oggetto `clip` filmato contenente l'istanza video.
- La proprietà `_rotation` della classe `Video`, che consente di impostare il grado di rotazione dell'istanza video.
- La proprietà `_visible` della classe `Video`, che consente di impostare la visibilità di un'istanza video.
- La proprietà `_width` della classe `Video`, che consente di impostare la larghezza dell'istanza video.

- La proprietà `_x` della classe `Video`, che consente di impostare la coordinata *x* dell'istanza video.
- La proprietà `_xmouse` della classe `Video`, che consente di impostare la coordinata *x* della posizione del puntatore del mouse.
- La proprietà `_xscale` della classe `Video`, che consente di impostare la scala orizzontale, espressa in percentuale, dell'istanza video.
- La proprietà `_y` della classe `Video`, che consente di impostare la coordinata *y* dell'istanza video.
- La proprietà `_ymouse` della classe `Video`, che consente di impostare la coordinata *y* della posizione del puntatore del mouse.
- La proprietà `_yscale` della classe `Video`, che consente di impostare la scala verticale, espressa in percentuale, dell'istanza video.
- Il gestore di eventi `onHTTPStatus` della classe `XML` restituisce il codice di stato inviato dal server (per esempio, il valore 404 per una pagina non trovata). Per ulteriori informazioni, vedere `onHTTPStatus` (gestore `XML.onHTTPStatus`) nella *Guida di riferimento di ActionScript 2.0*.
- La proprietà `localName` della classe `XMLNode`, che restituisce il nome completo dell'oggetto nodo XML (compresi prefisso e nome locale).
- La proprietà `namespaceURI` della classe `XMLNode`, che legge l'URI dello spazio dei nomi in cui si risolve il prefisso del nodo XML. Per ulteriori informazioni, vedere `namespaceURI` (proprietà `XMLNode.namespaceURI`) nella *Guida di riferimento di ActionScript 2.0*.
- La proprietà `prefix` della classe `XMLNode`, che legge il prefisso del nome del nodo.
- Il metodo `getNamespaceForPrefix()` della classe `XMLNode`, che restituisce l'URI dello spazio dei nomi associato al prefisso specificato per il nodo.
- Il metodo `getPrefixForNamespace` della classe `XMLNode`, che restituisce il prefisso associato a un URI dello spazio dei nomi specificato per il nodo.

Informazioni sugli elementi di linguaggio sconsigliati

In Flash Player 8 alcuni elementi del linguaggio sono sconsigliati. Per un elenco degli elementi di linguaggio sconsigliati e delle alternative da utilizzare in Flash Player 8, vedere le sezioni seguenti nella *Guida di riferimento di ActionScript 2.0*:

- [Riepilogo delle classi obsolete](#)
- [Riepilogo delle funzioni obsolete](#)
- [Riepilogo delle proprietà obsolete](#)

- Riepilogo degli operatori obsoleti

Modifiche all'Editor di ActionScript

L'editor di ActionScript del pannello Azioni e della finestra Script è stato aggiornato in diversi modi così da renderlo più affidabile e facile da utilizzare rispetto alle versioni precedenti dello strumento. Le modifiche sono riepilogate in questa sezione.

Visualizzazione dei caratteri nascosti È ora possibile usare il menu a comparsa Opzioni nel riquadro Script, nel pannello Debugger e nel pannello Output, per visualizzare o nascondere i caratteri nascosti quando si scrivono file di script nel pannello Azioni o nella finestra Script. Per informazioni su questa funzione, vedere “[Visualizzazione dei caratteri nascosti](#)” a pagina 58.

Aggiunta di assistente Script al pannello Azioni Nelle versioni precedenti di Flash era possibile utilizzare il pannello Azioni in *modalità normale*, immettendo opzioni e parametri per creare il codice, oppure in *modalità esperto*, aggiungendo i comandi direttamente nel riquadro Script. Queste opzioni non erano disponibili in Flash MX 2004 e Flash MX Professional 2004. Tuttavia, in Flash Basic 8 e Flash Professional 8, è possibile lavorare in *modalità Assistente script*, cioè in una modalità simile a quella normale ma più affidabile. Per informazioni sull'Assistente script, vedere [Capitolo 13, “Creazione di script di ActionScript mediante Assistente script”](#) nella guida *Uso di Flash*. Per un'esercitazione sull'Assistente script, vedere [Capitolo 13, “Creazione di un evento startDrag/stopDrag mediante Assistente script”](#) nella guida *Uso di Flash*.

Ricarica di file modificati Quando si lavora su un'applicazione è possibile ricaricare i file di script. Appare un messaggio di avvertimento che avvisa di ricaricare i file di script modificati associati all'applicazione a cui si sta lavorando. Questa funzione è particolarmente utile per i team che lavorano contemporaneamente sulle applicazioni, poiché impedisce di lavorare con script vecchi o di sovrascrivere le versioni più recenti di uno script. Se un file di script è stato spostato o cancellato, appare un messaggio di avvertimento che avvisa di salvare i file quando necessario. Per ulteriori informazioni, vedere “[Informazioni sulle preferenze di ActionScript](#)” a pagina 45.

Cambiamenti del modello di sicurezza dei file SWF installati localmente

Flash Player 8 dispone di un nuovo modello di sicurezza migliorato in cui le applicazioni Flash e i file SWF presenti su un computer locale possono comunicare con Internet e con il file system locale, invece di essere eseguiti da un server Web remoto. Quando si sviluppa un'applicazione Flash, è necessario indicare se a un file SWF è consentito comunicare con una rete o con un file system locale.



In questa descrizione, per *file SWF locale* si intende un file SWF installato localmente sul computer di un utente e non servito da un sito Web, che non include file di proiettore (EXE).

Nelle versioni precedenti di Flash Player, i file SWF locali potevano interagire con altri file SWF e caricare i dati da qualsiasi computer remoto o locale senza configurare le impostazioni di sicurezza. In Flash Player 8, un file SWF non può eseguire connessioni al file system locale e alla rete (come Internet) nella stessa applicazione senza configurare un'impostazione di sicurezza. In questo modo si aumenta la sicurezza: un file SWF non può leggere i file presenti sul disco rigido e inviare i contenuti di tali file attraverso Internet.

Questa restrizione di sicurezza influisce solo sul contenuto distribuito localmente, che sia contenuto precedente (un file FLA creato con un versione antecedente di Flash) o creato in Flash 8. Utilizzando Flash MX 2004 o uno strumento di creazione precedente, è possibile verificare un'applicazione Flash eseguita localmente e in grado di accedere a Internet. In Flash Player 8 viene chiesta all'utente l'autorizzazione esplicita per la comunicazione con Internet.

Quando si prova un file presente sul disco rigido, esistono diversi passaggi per stabilire se il file è un documento locale affidabile (sicuro) o un documento potenzialmente pericoloso (non sicuro). Se si crea il file nell'ambiente di creazione Flash (per esempio, quando si seleziona Controllo > Prova filmato), il file è affidabile perché si trova nell'ambiente di prova.

In Flash Player 7 e nelle versioni precedenti, i file SWF locali disponevano di autorizzazioni che consentivano di accedere sia al file system locale che alla rete. In Flash Player 8, i file SWF locali possono avere tre diversi livelli di autorizzazione:

- Accesso solo al file system locale (livello predefinito). Il file SWF locale può leggere dal file system locale e dai percorsi di rete UNC (Universal Naming Convention) e non può comunicare con Internet.
- Accesso solo alla rete. Il file SWF locale può accedere solo alla rete (come Internet) e non al file system locale in cui è installato.
- Accesso sia alla rete che al file system locale. Il file SWF locale può leggere dal file system locale in cui è installato, leggere e scrivere su qualsiasi server a cui può accedere e può eseguire script su altri file SWF presenti in rete o nel file system locale a cui può accedere.

Per ulteriori dettagli su ciascun livello di autorizzazione, vedere “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

Sono state anche eseguite piccole modifiche a System.security.allowDomain e miglioramenti a System.security.allowInsecureDomain. Per ulteriori informazioni sulla sicurezza dei file locali, vedere [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)

CAPITOLO 2

Scrittura e modifica del codice ActionScript 2.0

Il codice ActionScript in Macromedia Flash Basic 8 o Macromedia Flash Professional 8 si scrive utilizzando il pannello Azioni o la finestra Script, che offrono un editor di codice completamente funzionale (chiamato *Editor di ActionScript*) provvisto di suggerimenti sul codice, colorazione, formattazione, evidenziazione e controllo della sintassi, debug, numeri di riga, funzione di a capo automatico e supporto di Unicode in due viste diverse. Per ulteriori informazioni sull'Editor di ActionScript, vedere “[Uso del pannello Azioni e della finestra Script](#)” a pagina 36.

Per scrivere codice ActionScript in Flash è possibile utilizzare due metodi: è possibile scrivere script che fanno parte del documento Flash (vale a dire script incorporati nel file FLA), oppure script esterni (script o classi memorizzati in file esterni). Per scrivere script esterni non è possibile utilizzare il pannello Azioni.

Gli script in un file FLA si scrivono con l'Editor di ActionScript nel pannello Azioni, che contiene l'Editor di ActionScript in un riquadro Script e strumenti di supporto per facilitare la scrittura di script. Il pannello Azioni contiene la casella degli strumenti Azioni (che consente di accedere rapidamente agli elementi principali del linguaggio ActionScript), Script Navigator (che consente di navigare fra tutti gli script del proprio documento) e Assistente script (che guida l'utente richiedendo l'immissione degli elementi necessari alla creazione degli script). Per ulteriori informazioni sul pannello Azioni, vedere “[Informazioni sul pannello Azioni](#)” a pagina 37. Per ulteriori informazioni su Assistente script, vedere “[Informazioni su Assistente script](#)” a pagina 63.

Quando è necessario creare uno script esterno, per la creazione di un nuovo file AS è possibile usare l'Editor di ActionScript nella finestra Script. Per la creazione di un file AS esterno è anche possibile utilizzare il proprio editor di testo preferito. Nella finestra Script l'Editor di ActionScript contiene funzioni di assistenza quali i suggerimenti sul codice, la colorazione, la verifica della sintassi e così via, come per il pannello Azioni. Per ulteriori informazioni sulla finestra Script, vedere “[Informazioni sulla finestra Script](#)” a pagina 38.

Flash assiste ulteriormente l'utente nella creazione degli script mediante i comportamenti, funzioni predefinite di ActionScript che possono essere associate agli oggetti del documento Flash senza che sia necessario creare il codice ActionScript. Per ulteriori informazioni sui comportamenti, vedere “[Informazioni sui comportamenti](#)” a pagina 66.

Per ulteriori informazioni sulla gestione degli eventi, consultare le seguenti sezioni:

Informazioni su ActionScript ed eventi	32
Organizzazione del codice ActionScript	34
Uso del pannello Azioni e della finestra Script	36
Informazioni sul pannello Azioni	37
Informazioni sulla finestra Script	38
Informazioni sulla creazione di codice nel pannello Azioni e nella finestra Script	40
Informazioni sulle funzioni del pannello Azioni	62
Informazioni sui comportamenti	66
Informazioni sulle impostazioni di pubblicazione ActionScript	67

Informazioni su ActionScript ed eventi

In Macromedia Flash Basic 8 e Macromedia Flash Professional 8, il codice ActionScript viene eseguito quando si verifica un evento: ad esempio, quando viene caricato un clip filmato, quando viene immesso un fotogramma nella linea temporale, o quando l'utente fa clic su un pulsante. Gli eventi possono essere attivati dall'utente o dal sistema. Gli utenti fanno clic sul pulsante del mouse e premono i tasti, mentre il sistema attiva gli eventi quando vengono soddisfatte determinate condizioni o vengono completati dei processi (un file SWF viene caricato, la linea temporale raggiunge un determinato fotogramma, viene terminato lo scaricamento di un'immagine e così via).

Quando si verifica un evento, l'utente crea un *gestore di eventi* che risponde con una determinata azione. Comprendere quando e dove si verificano gli eventi consente di determinare come e quando è necessario rispondere agli stessi mediante un'azione, e quale strumento di ActionScript in Flash 8 deve essere utilizzato nei singoli casi. Per ulteriori informazioni, consultare “[Informazioni sulla creazione di script per la gestione degli eventi](#)” a pagina 35.

Gli eventi possono essere suddivisi in diverse categorie: eventi di mouse e di tastiera, che si verificano quando un utente interagisce con l'applicazione mediante il mouse e la tastiera; eventi di clip filmato, che si verificano in clip filmato ed eventi di fotogramma, che si verificano nei fotogrammi sulla linea temporale.

Per ulteriori informazioni sui tipi di script che si possono scrivere per la gestione di eventi, vedere “[Informazioni sulla creazione di script per la gestione degli eventi](#)” a pagina 35

Eventi associati a mouse e tastiera

Un utente che interagisce con il file SWF o l'applicazione Flash attiva eventi del mouse e della tastiera. Ad esempio, quando si passa il puntatore sopra un pulsante, si verifica l'evento `Button.onRollOver` o `on(rollOver)`; quando viene premuto un pulsante, si verifica l'evento `Button.onRelease`; se viene premuto un tasto della tastiera, si verifica l'evento `on(keyPress)`. È possibile scrivere codice su un fotogramma o allegare a un'istanza degli script che gestiscano questi eventi e aggiungervi gli elementi di interattività desiderati.

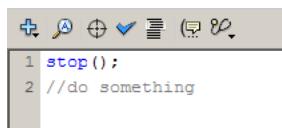
Eventi associati ai clip filmato

All'interno di un clip filmato, è possibile reagire a una serie di eventi clip che vengono attivati quando l'utente entra o esce dalla scena o interagisce con la scena mediante il mouse o la tastiera. Ad esempio, è possibile caricare un file SWF o un'immagine JPG esterni nel clip filmato nel momento in cui inizia la riproduzione di una scena, o fare in modo che sia possibile spostare gli elementi della scena mediante i movimenti del mouse.

Eventi associati ai fotogrammi

In una linea temporale principale o nella linea temporale di un clip filmato, un evento di sistema si verifica nel momento in cui l'indicatore di riproduzione entra in un fotogramma chiave; questo tipo di evento è noto come *evento fotogramma*. Gli eventi fotogramma sono utili per attivare le azioni in base al passare del tempo (lo spostamento attraverso la linea temporale) o per interagire con gli elementi attualmente visibili sullo stage. Quando a un fotogramma chiave si aggiunge uno script, questo viene eseguito nel momento in cui la riproduzione raggiunge il fotogramma chiave. Uno script associato a un fotogramma si definisce *script di fotogramma*.

Gli script di fotogramma sono più comunemente utilizzati per arrestare la riproduzione nel momento in cui si raggiunge un determinato fotogramma chiave. Questo avviene mediante la funzione `stop()`. Nel pannello Azioni, selezionare un fotogramma chiave, quindi aggiungere la funzione `stop()` come elemento dello script.

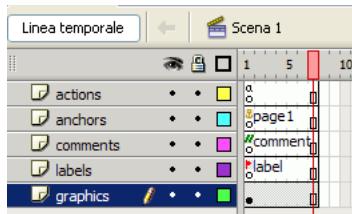


Dopo aver arrestato il file SWF in corrispondenza di un determinato fotogramma chiave, è necessario eseguire delle operazioni; ad esempio, usare uno script di fotogramma per aggiornare in modo dinamico il valore di un'etichetta, per gestire l'interazione degli elementi sullo stage e così via.

Organizzazione del codice ActionScript

Gli script possono essere associati ai fotogrammi chiave e alle istanze di oggetto (clip filmato, pulsanti e altri simboli). Tuttavia, se il codice ActionScript è distribuito su più fotogrammi chiave e istanze di oggetto, il debugging dell'applicazione risulterà molto più difficile e sarà difficile anche condividere il codice tra più applicazioni Flash. Per questi motivi è importante seguire le procedure consigliate relative alla creazione di codice ActionScript in Flash.

Invece di associare gli script ad elementi, quali fotogrammi chiave, clip filmato e pulsanti, si consiglia di rispondere agli eventi chiamando delle funzioni che risiedono in una posizione centrale. Un metodo possibile consiste nell'associare il codice ActionScript incorporato al primo o al secondo fotogramma di una linea temporale, se possibile, in modo da non doverlo cercare in tutto il file FLA. Un metodo comunemente utilizzato consiste nel creare un livello denominato *actions* e inserirvi il codice ActionScript.



Quando si associano tutti gli script ai singoli elementi, si incorpora tutto il codice nel file FLA. Se è necessario condividere il codice con altre applicazioni Flash, usare la finestra Script o l'editor di testo preferito per creare un file ActionScript (AS) esterno.

La creazione di un file esterno consente di organizzare meglio il codice rendendolo più modulare; una funzione tanto più utile quanto più il progetto progredisce. Un file esterno facilita le operazioni di debug e la gestione del controllo dei codici di origine, se si lavora su un progetto con altri sviluppatori.

Per utilizzare il codice ActionScript contenuto in un file AS esterno, si crea uno script nel file FLA, quindi si usa l'istruzione `#include` per accedere al codice memorizzato esternamente, come nel seguente esempio:

```
#include "../core/Functions.as"
```

Il codice ActionScript 2.0 può essere utilizzato anche per creare classi personalizzate che devono essere memorizzate in file AS esterni. Per esportare le classi nel file SWF è necessario utilizzare istruzioni `import` anziché `#include`. Per ulteriori informazioni sulla scrittura di file di classe, vedere “[Creazione di file di classi personalizzate](#)” a pagina 208 e “[Informazioni sull'importazione di file di classe](#)” a pagina 212 relativi all'importazione di tali file. È anche possibile utilizzare componenti (clip filmato predefiniti) per condividere codice e funzionalità, quali elementi UI e script.

NOTA

Il codice ActionScript nei file esterni viene compilato in un file SWF al momento della pubblicazione, dell'esportazione, della verifica o del debug di un file FLA. Pertanto, in caso di modifiche a un file esterno, è necessario salvare il file e ricompilare gli eventuali file FLA che lo utilizzano.

La scrittura di codice ActionScript in Flash 8 avviene mediante il pannello Azioni, la finestra Script, o entrambi. La scelta di utilizzare uno dei due metodi è determinata dalla modalità di risposta agli eventi, da come viene organizzato il codice e, soprattutto, dalle procedure consigliate e più indicate per i singoli casi.

Per ulteriori informazioni sulle procedure consigliate e sulle convenzioni di codifica, vedere “[Convenzioni di codifica ActionScript](#)” a pagina 807.

Quando si utilizzano i comportamenti, funzioni predefinite di ActionScript (vedere “[Informazioni sui comportamenti](#)” a pagina 66), devono essere prese in considerazione altre questioni sul flusso di lavoro e sull'organizzazione del codice.

Informazioni sulla creazione di script per la gestione degli eventi

La scrittura di codice per gli eventi può essere suddivisa in due categorie principali: eventi che si verificano sulla linea temporale (nei fotogrammi chiave) ed eventi che si verificano sulle istanze di oggetti (clip filmato, pulsanti e componenti). L'interattività dell'applicazione o del file SWF può essere distribuita fra i molti elementi del progetto e, se si desidera, è possibile aggiungere degli script direttamente agli elementi. Tuttavia, si consiglia di non aggiungere gli script direttamente agli elementi (oggetti e fotogrammi chiave), ma di rispondere agli eventi chiamando delle funzioni che risiedono in una posizione centrale, come descritto in “[Organizzazione del codice ActionScript](#)”.

Uso del pannello Azioni e della finestra Script

Per creare script in un file FLA, immettere il codice ActionScript direttamente nel pannello Azioni. Per creare script esterni da includere o importare nell'applicazione, utilizzare l'editor di testo desiderato o la finestra Script (File > Nuovo, quindi scegliere File ActionScript).

Quando si usa il pannello Azioni o la finestra Script, si utilizzano le funzioni dell'Editor di ActionScript per la scrittura, la formattazione e la modifica del codice; sia il pannello Azioni che la finestra Script dispongono del riquadro Script (dove viene scritto il codice) e della casella degli strumenti Azioni. Il pannello Azioni offre più funzioni di assistenza rispetto alla finestra Script; tali funzioni si trovano nel pannello Azioni perché sono utili in particolare nel contesto della modifica di ActionScript in un file FLA.

Per visualizzare il pannello Azioni, effettuare una delle seguenti operazioni:

- Selezionare Finestra > Azioni.
- Premere F9.

Per visualizzare la finestra Script, effettuare una delle seguenti operazioni:

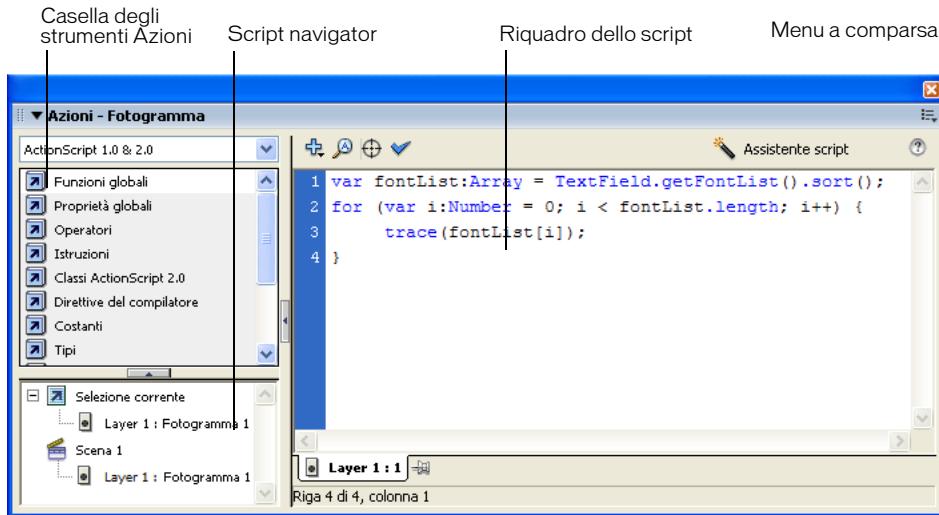
- Per iniziare la creazione di un nuovo script, selezionare File > Nuovo, quindi scegliere File ActionScript.
- Per aprire uno script esistente, selezionare File > Apri e aprire un file di ActionScript (AS) esistente.
- Per modificare uno script già aperto, fare clic sulla scheda del documento in cui è visualizzato il nome dello script.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Informazioni sul pannello Azioni](#)” a pagina 37
- “[Informazioni sulla finestra Script](#)” a pagina 38

Informazioni sul pannello Azioni

Il pannello Azioni consente di creare il codice ActionScript in un documento Flash (un file FLA), ed è suddiviso in tre riquadri, ognuno predisposto per aiutare l'utente durante la creazione e la gestione degli script.



Casella degli strumenti Azioni Utilizzare la casella degli strumenti Azioni per sfogliare l'elenco degli elementi del linguaggio ActionScript suddivisi per categorie (funzioni, classi, tipi e così via), quindi inserirli nel riquadro Script. Per inserire un elemento dello script nel riquadro Script, fare doppio clic sull'elemento o trascinarlo direttamente nel riquadro. È anche possibile aggiungere elementi del linguaggio agli script mediante il pulsante Aggiungi (+) nella barra degli strumenti del pannello Azioni. Per ulteriori informazioni, vedere “[Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script](#)” a pagina 41.

Script navigator Script navigator visualizza un elenco gerarchico di elementi Flash (clip filmato, fotogrammi e pulsanti) che contengono gli script. e consente di spostarsi rapidamente fra tutti gli script del documento Flash.

Se si fa clic su un elemento in Script navigator, lo script associato all'elemento viene visualizzato nel riquadro Script, mentre l'indicatore di riproduzione si sposta sulla posizione corrispondente nella linea temporale. Se si fa doppio clic su un elemento in Script navigator, lo script viene *bloccato*. Per ulteriori informazioni, vedere “[Blocco degli script nel pannello Azioni](#)” a pagina 63.

Riquadro Script Il codice viene scritto nel riquadro Script, che offre gli strumenti per creare gli script in un editor completamente funzionale (chiamato *Editor di ActionScript*) provvisto di formattazione e controllo della sintassi, suggerimenti sul codice, colorazione, debugging e altre funzioni che semplificano la creazione degli script. Per ulteriori informazioni, vedere “[Uso del pannello Azioni e della finestra Script](#)” a pagina 36.

Per ulteriori informazioni sui pulsanti nella barra degli strumenti del pannello Azioni, vedere “[Informazioni sulla creazione di codice nel pannello Azioni e nella finestra Script](#)” a pagina 40. Per ulteriori informazioni sulle funzioni del pannello Azioni, consultare i seguenti argomenti:

- “[Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script](#)” a pagina 41
- “[Informazioni sulle opzioni di modifica di ActionScript](#)” a pagina 43
- “[Informazioni sui suggerimenti sul codice in Flash](#)” a pagina 47
- “[Formattazione di codice](#)” a pagina 54
- “[Uso dell'evidenziazione della sintassi](#)” a pagina 55
- “[Uso di numeri di riga e di ritorno a capo automatico](#)” a pagina 56
- “[Uso dei tasti di scelta rapida Esc](#)” a pagina 57
- “[Visualizzazione dei caratteri nascosti](#)” a pagina 58
- “[Uso dello strumento Trova](#)” a pagina 59
- “[Controllo della sintassi e della punteggiatura](#)” a pagina 59
- “[Importazione ed esportazione degli script](#)” a pagina 60

Informazioni sulla finestra Script

Quando si crea un nuovo file ActionScript, un file di comunicazione Flash o un file JavaScript Flash, è possibile scrivere e modificare ActionScript nella finestra Script, utilizzata per scrivere e modificare file di script esterni. I colori per la sintassi, i suggerimenti sul codice e altre opzioni dell'Editor sono supportati nella finestra Script.

Nella finestra Script è possibile creare file ActionScript esterni, file di comunicazione ActionScript e file JavaScript Flash. A seconda del tipo di file di script esterno che si desidera creare, la casella degli strumenti Azioni offre un elenco completo degli elementi del linguaggio disponibili.

Quando si usa la finestra Script si può notare che alcune delle altre funzioni di assistenza sul codice, quali Script navigator, la modalità Assistente script e i comportamenti, non sono disponibili. Questo avviene perché alcune delle funzioni sono utili solo nell'ambito della creazione dei documenti Flash, non dei file di script esterni.

Si noterà inoltre che non sono disponibili anche molte delle opzioni solitamente disponibili nel pannello Azioni. Nella finestra Script sono supportate le opzioni dell'Editor seguenti: casella degli strumenti Azioni, trova e sostituisci, controllo della sintassi, formattazione automatica, suggerimenti sul codice e opzioni di debug (solo per file ActionScript). Inoltre, la finestra Script supporta la visualizzazione dei numeri di riga, dei caratteri nascosti e la funzione di a capo automatico.

Per visualizzare la finestra Script:

1. Selezionare File > Nuovo.
2. Selezionare il tipo di file esterni che si desidera creare (file ActionScript, file comunicazione ActionScript o file JavaScript Flash).

È possibile aprire contemporaneamente più file esterni; i nomi dei file sono visualizzati nelle schede presenti nella sezione superiore della finestra Script Per ulteriori informazioni sulle funzioni della finestra Script, consultare i seguenti argomenti:

- “Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script” a pagina 41
- “Informazioni sulle opzioni di modifica di ActionScript” a pagina 43
- “Informazioni sui suggerimenti sul codice in Flash” a pagina 47
- “Formattazione di codice” a pagina 54
- “Uso dell'evidenziazione della sintassi” a pagina 55
- “Uso di numeri di riga e di ritorno a capo automatico” a pagina 56
- “Uso dei tasti di scelta rapida Esc” a pagina 57
- “Visualizzazione dei caratteri nascosti” a pagina 58
- “Uso dello strumento Trova” a pagina 59
- “Controllo della sintassi e della punteggiatura” a pagina 59
- “Importazione ed esportazione degli script” a pagina 60

Informazioni sulla creazione di codice nel pannello Azioni e nella finestra Script

Il riquadro Script, in cui si modifica il codice, è l'elemento principale del pannello Azioni e della finestra Script e fornisce funzioni basilari di modifica degli script e assistenza sul codice quali i suggerimenti, la colorazione, la formattazione automatica e così via.

Le funzioni di assistenza alla modifica del codice si trovano nella barra degli strumenti del pannello Azioni o della finestra Script, nel sistema di menu e nel riquadro Script.

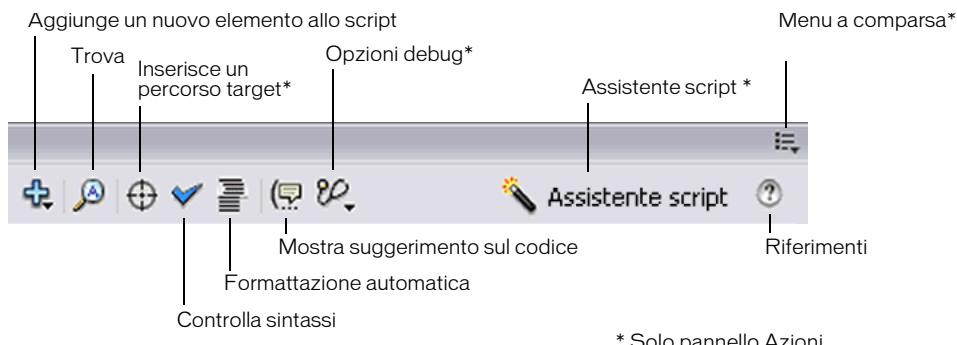
I seguenti argomenti descrivono le molteplici funzioni dell'Editor di ActionScript (pannello Azioni e finestra Script):

- “[Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script](#)” a pagina 41
- “[Informazioni sulle opzioni di modifica di ActionScript](#)” a pagina 43
- “[Informazioni sulle preferenze di ActionScript](#)” a pagina 45
- “[Informazioni sui suggerimenti sul codice in Flash](#)” a pagina 47
- “[Formattazione di codice](#)” a pagina 54
- “[Uso dell'evidenziazione della sintassi](#)” a pagina 55
- “[Uso di numeri di riga e di ritorno a capo automatico](#)” a pagina 56
- “[Uso dei tasti di scelta rapida Esc](#)” a pagina 57
- “[Visualizzazione dei caratteri nascosti](#)” a pagina 58
- “[Uso dello strumento Trova](#)” a pagina 59
- “[Controllo della sintassi e della punteggiatura](#)” a pagina 59
- “[Importazione ed esportazione degli script](#)” a pagina 60

Per funzioni specifiche del pannello Azioni, quali blocco degli script e Script navigator, vedere “[Informazioni sulle funzioni del pannello Azioni](#)” a pagina 62

Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script

Le barre degli strumenti dell'Editor di ActionScript e della finestra Script contengono i collegamenti alle funzioni di assistenza sul codice che semplificano e consentono di organizzare meglio la creazione di codice in ActionScript. Le barre degli strumenti sono diverse, a seconda che si utilizzi l'Editor di ActionScript nel pannello Azioni o il riquadro Script. L'immagine seguente visualizza le funzioni della barra degli strumenti del pannello Azioni: le opzioni contrassegnate sono disponibili solo nel pannello Azioni.



Le funzioni della barra degli strumenti sono trattate in dettaglio in “[Uso del pannello Azioni e della finestra Script](#)” a pagina 36. Di seguito viene fornito un breve riepilogo dei pulsanti disponibili nelle barre degli strumenti del pannello Azioni e della finestra Script.



Alcune delle seguenti opzioni sono disponibili solo nel pannello Azioni e sono contrassegnate come *Solo pannello Azioni*.

Aggiunge un nuovo elemento allo script Visualizza tutti gli elementi del linguaggio che si trovano anche nella casella degli strumenti di ActionScript. Selezionare un elemento dall'elenco degli elementi del linguaggio per aggiungerlo allo script.

Trova Trova e sostituisce il testo nel codice ActionScript. Per ulteriori informazioni, vedere “[Uso dello strumento Trova](#)” a pagina 59.

Inserisce un percorso target *Solo pannello Azioni*. Fornisce assistenza durante l'impostazione di un percorso target assoluto o relativo per un'azione dello script. Per ulteriori informazioni, vedere “[Inserimento dei percorsi target](#)” a pagina 65.

Controlla sintassi Verifica la presenza di eventuali errori di sintassi nello script corrente. Gli errori di sintassi vengono elencati nel pannello Output. Per ulteriori informazioni, vedere “[Controllo della sintassi e della punteggiatura](#)” a pagina 59.

Formattazione automatica Formatta lo script per garantire un'adeguata sintassi del codice e migliorare la leggibilità. È possibile impostare le preferenze di formattazione automatica nella finestra di dialogo Preferenze, accessibile mediante il menu Modifica o mediante il menu a comparsa del pannello Azioni. Per ulteriori informazioni, vedere “[Formattazione di codice](#)” a pagina 54.

Mostra suggerimento sul codice Se è stata disattivata la funzione di suggerimento sul codice, è possibile usare questo comando per visualizzare manualmente un suggerimento relativo alla riga di codice su cui si sta lavorando. Per ulteriori informazioni, vedere “[Informazioni su Assistente script](#)” a pagina 63.

Opzioni debug Imposta e rimuove i punti di interruzione nello script, in modo che durante il debug del documento Flash sia possibile arrestare l'operazione e procedere riga per riga nello script. Ora le opzioni di debug sono disponibili sia nella finestra Script che nel pannello Azioni, ma solo per i file di ActionScript; l'opzione è disabilitata per i file comunicazione di ActionScript e per i file JavaScript Flash. Per ulteriori informazioni sulle operazioni di debug dei documenti Flash, vedere “[Esecuzione del debug degli script](#)” a pagina 771. Per informazioni sull'impostazione e la rimozione di punti di interruzione, vedere “[Impostazione e rimozione dei punti di interruzione](#)” a pagina 780.

Assistente script Solo pannello Azioni. La modalità Assistente script richiede l'immissione degli elementi necessari a creare gli script. Per ulteriori informazioni, vedere “[Informazioni su Assistente script](#)” a pagina 63.

Guida di riferimento Visualizza un argomento della Guida di riferimento relativo all'elemento del linguaggio ActionScript selezionato nel riquadro Script. Ad esempio, se si fa clic su un'istruzione `import` quindi si seleziona il pulsante della Guida di riferimento, nel pannello della Guida viene visualizzato l'argomento relativo all'`importazione`.

Menu a comparsa Solo pannello Azioni. Contiene i comandi e le preferenze per il pannello Azioni o per la finestra Script. Ad esempio, è possibile impostare i numeri di riga e la funzione di a capo automatico nell'Editor di ActionScript, accedere alle preferenze di ActionScript e importare o esportare gli script. Per ulteriori informazioni, vedere “[Informazioni sulle opzioni di modifica di ActionScript](#)” a pagina 43.

Informazioni sulle opzioni di modifica di ActionScript

La finestra Script e il pannello Azioni offrono molte funzioni di assistenza relative al codice, vale a dire, una serie di strumenti che semplificano molto la scrittura e la gestione degli script. Tali strumenti sono disponibili nella barra degli strumenti del pannello Azioni o della finestra Script e nel menu a comparsa del pannello Azioni. Durante la modifica di ActionScript nella finestra Script, queste opzioni sono disponibili nella barra degli strumenti e nei menu di Flash.

Il pannello Azioni fornisce un numero maggiore di opzioni rispetto a quelle che sono disponibili nella finestra Script; queste opzioni aggiuntive sono utili per la creazione di codice ActionScript incorporato nei documenti Flash, non durante la scrittura di file ActionScript esterni. Per ulteriori informazioni sulle opzioni disponibili nella finestra Script, vedere [“Informazioni sulla finestra Script” a pagina 38](#).

Le opzioni disponibili nella finestra Script e nel pannello Azioni sono trattate in [“Informazioni sulle barre degli strumenti del pannello Azioni e della finestra Script” a pagina 41](#).

Le seguenti opzioni sono disponibili nel menu a comparsa del pannello Azioni e in diversi altri menu della finestra Script.



Alcune delle seguenti opzioni sono disponibili solo nel pannello Azioni e sono contrassegnate come *Solo pannello Azioni*.

Ricarica suggerimenti codice *Solo pannello Azioni*. Se si personalizza la modalità

Assistente script scrivendo metodi personalizzati, è possibile ricaricare i suggerimenti di codice senza riavviare Flash 8.

Blocca lo script *Solo pannello Azioni*. Blocca in posizione lo script attualmente visualizzato nel pannello Script. Per ulteriori informazioni, vedere [“Blocco degli script nel pannello Azioni” a pagina 63](#).

Chiudi script *Solo pannello Azioni*. Chiude lo script attualmente aperto.

Chiudi tutti gli script *Solo pannello Azioni*. Chiude tutti gli script attualmente aperti.

Vai alla riga Individua ed evidenzia la riga specificata nel riquadro Script.

Trova e sostituisci Trova e sostituisce il testo all'interno degli script nel riquadro Script. Per ulteriori informazioni, vedere [“Uso dello strumento Trova” a pagina 59](#).

Trova di nuovo Ripete l'operazione di ricerca dell'ultima stringa immessa nello strumento Trova. Per ulteriori informazioni, vedere [“Uso dello strumento Trova” a pagina 59](#).

Importa script Consente di importare un file di script (ActionScript) nel riquadro Script. Per ulteriori informazioni, vedere [“Importazione ed esportazione delle preferenze” a pagina 61](#).

Esporta script Esporta lo script attuale in un file ActionScript (AS) esterno. Per ulteriori informazioni, vedere “[Importazione ed esportazione delle preferenze](#)” a pagina 61.

Tasti di scelta rapida Esc Immette negli script elementi di linguaggio e strutture di sintassi comuni. Ad esempio, se si digita Esc+g+p nel riquadro Script, la funzione `gotoAndPlay()` viene inserita nello script. Quando si seleziona l’opzione Tasti di scelta rapida Esc nel menu a comparsa del pannello Azioni, tutti i tasti di scelta rapida escape disponibili vengono visualizzati nella casella degli strumenti Azioni. Per ulteriori informazioni, vedere “[Uso dei tasti di scelta rapida Esc](#)” a pagina 57.

Caratteri nascosti Visualizza i caratteri nascosti nello script, vale a dire gli spazi, le tabulazioni e le interruzioni di riga. Per ulteriori informazioni, vedere “[Visualizzazione dei caratteri nascosti](#)” a pagina 58.

Numeri di riga Visualizza i numeri di riga nel riquadro Script. Per ulteriori informazioni, vedere “[Uso di numeri di riga e di ritorno a capo automatico](#)” a pagina 56.

Preferenze *Solo pannello Azioni.* Visualizza la finestra di dialogo Preferenze di ActionScript. Per ulteriori informazioni, vedere “[Informazioni sulle preferenze di ActionScript](#)” a pagina 45.

A capo automatico Per fare in modo che le righe dello script più lunghe della finestra Script vadano a capo, selezionare l’opzione A capo automatico dal menu a comparsa del pannello Azioni. Quando si utilizza la finestra Script, selezionare A capo automatico dal menu Visualizza. Per ulteriori informazioni, vedere “[Uso di numeri di riga e di ritorno a capo automatico](#)” a pagina 56.

Raggruppa Azioni con *Solo pannello Azioni.* Consente di raggruppare il pannello Azioni, che comprende la barra degli strumenti Azioni e lo Script navigator, con altri pannelli dell’ambiente di creazione Flash.

Inoltre, il menu a comparsa del pannello Azioni include i comandi Stampa e ? (Guida) e i comandi di ridimensionamento dei pannelli.

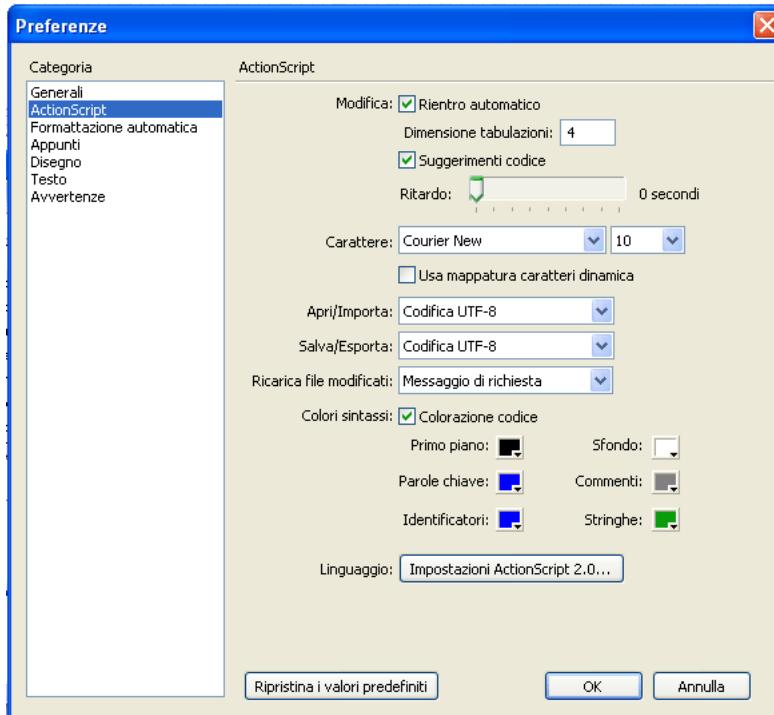
Informazioni sulle preferenze di ActionScript

Che si modifichi il codice nel pannello Azioni o nella finestra Script, è possibile modificare un'unica serie di preferenze. Ad esempio, è possibile controllare la funzione di rientro automatico, i suggerimenti sul codice, la colorazione e una serie di altre funzioni basilari di modifica del codice.

Per accedere alle preferenze di ActionScript:

1. Per accedere alle preferenze ActionScript in un file FLA con il pannello Azioni, selezionare Preferenze dal menu a comparsa o Modifica > Preferenze (Windows), oppure Flash > Preferenze (Macintosh) e selezionare ActionScript dall'elenco Categoria.
2. Per accedere alle preferenze ActionScript nella finestra Script, selezionare Modifica > Preferenze, quindi scegliere ActionScript (Windows) o Flash > Preferenze e selezionare ActionScript (Macintosh).

La seguente immagine mostra le impostazioni ActionScript che possono essere modificate in Flash 8.



È possibile impostare le seguenti preferenze:

Rientro automatico Quando il rientro automatico è attivo, al testo digitato dopo un segno di parentesi aperta [(o {} viene automaticamente applicato un rientro in base alle impostazioni definite in Dimensione tabulazioni nelle preferenze di ActionScript. Per ulteriori informazioni, vedere “[Formattazione di codice](#)” a pagina 54.

Dimensione tabulazioni Specifica il numero di caratteri di rientro della nuova riga quando è attiva l'opzione di rientro automatico.

Suggerimenti codice Abilita la funzione dei suggerimenti sul codice nel riquadro Script. Per ulteriori informazioni sull'uso dei suggerimenti sul codice, vedere “[Informazioni sui suggerimenti sul codice in Flash](#)” a pagina 47.

Ritardo Specifica il lasso di tempo, in secondi, prima che vengano visualizzati i suggerimenti sul codice.

Carattere Specifica il tipo di carattere usato nel riquadro Script.

Usa mappatura caratteri dinamica Verifica che la famiglia di caratteri selezionata disponga dei glifi necessari per il rendering di tutti i caratteri. Se la verifica dà esito negativo, Flash sostituisce questa famiglia con una che contenga tutti i caratteri necessari. Per ulteriori informazioni, vedere “[Formattazione di codice](#)” a pagina 54.

Codifica Specifica la codifica dei caratteri usata per l'apertura, il salvataggio, l'importazione e l'esportazione dei file ActionScript. Per ulteriori informazioni, vedere “[Importazione ed esportazione degli script](#)” a pagina 60.

Ricarica file modificati Consente di scegliere quando visualizzare messaggi di avvertimento se un file di script viene modificato, spostato o cancellato. Scegliere tra Sempre, Mai o Messaggio di richiesta.

- **Sempre** Quando viene rilevata una modifica, non viene visualizzato alcun avvertimento e il file viene ricaricato automaticamente.
- **Mai** Quando viene rilevata una modifica, non viene visualizzato alcun avvertimento e il file resta nello stato corrente.
- **Messaggio di richiesta** (Predefinito) Quando viene rilevata una modifica, viene visualizzato un avvertimento ed è possibile scegliere di ricaricare il file o di non ricaricarlo.

Quando si creano applicazioni che coinvolgono file di script esterni, questa funzione consente di evitare di sovrascrivere uno script modificato da un membro del team dopo l'apertura dell'applicazione, o di pubblicare l'applicazione con versioni di script datate. Gli avvertimenti consentono di chiudere automaticamente uno script e di riaprire la versione modificata più recente.

Colori sintassi Specifica i colori per la colorazione del codice negli script. Con la colorazione codice abilitata è possibile selezionare i colori da visualizzare nel riquadro Script.

Linguaggio Apre la finestra di dialogo Impostazioni ActionScript. Per ulteriori informazioni, vedere “[Modifica del percorso di classe](#)” a pagina 68.

Informazioni sui suggerimenti sul codice in Flash

Quando si usa il pannello Azioni o la finestra Script, è possibile servirsi di diverse funzioni che facilitano la scrittura di codice corretto dal punto di vista sintattico. I suggerimenti sul codice aiutano l'utente a scrivere il codice in modo rapido e accurato. La funzione dei suggerimenti sul codice comprende descrizioni dei comandi che contengono l'esatta sintassi e i menu che consentono di selezionare i nomi di metodi e proprietà. Le sezioni seguenti indicano come scrivere codice che utilizza queste funzioni.

- “[Informazioni sull'attivazione dei suggerimenti sul codice](#)” a pagina 47
- “[Uso dei suggerimenti sul codice](#)” a pagina 48
- “[Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice](#)” a pagina 51
- “[Informazioni sull'uso dei suffissi per attivare i suggerimenti sul codice](#)” a pagina 52
- “[Informazioni sull'uso dei commenti per attivare i suggerimenti sul codice](#)” a pagina 53

Informazioni sull'attivazione dei suggerimenti sul codice

Quando si lavora nel pannello Azioni o nella finestra Script, Flash è in grado di rilevare il tipo di azione che si sta eseguendo e visualizzare un *suggerimento sul codice*. Esistono due tipi diversi di suggerimenti: le descrizioni dei comandi, che contengono la sintassi completa di quell'azione, e un menu a comparsa nel quale sono elencati i nomi di proprietà e metodi possibili (definito anche forma di *completamento del codice*). Il menu a comparsa viene visualizzato per i parametri, le proprietà e gli eventi quando si usa la tipizzazione forte dei dati o si assegna un nome agli oggetti, come descritto successivamente in questa sezione.

I suggerimenti sul codice talvolta vengono visualizzati se si fa doppio clic su un elemento nella casella degli strumenti Azioni oppure si seleziona il pulsante Aggiungi (+) nella barra degli strumenti del pannello Azioni o della finestra Script per aggiungere azioni nel riquadro Script. Per informazioni su come utilizzare i suggerimenti sul codice quando vengono visualizzati, vedere “[Uso dei suggerimenti sul codice](#)” a pagina 48.

NOTA

I suggerimenti sul codice vengono attivati automaticamente per le classi native che non richiedono la creazione e la denominazione di un'istanza della classe, ad esempio Math, Key, Mouse e così via.

Per assicurarsi che i suggerimenti sul codice siano attivi, l'opzione Suggerimenti codice deve essere selezionata nella finestra di dialogo delle preferenze di ActionScript. Per ulteriori informazioni, vedere “[Informazioni sul pannello Azioni](#)” a pagina 37.

Uso dei suggerimenti sul codice

I suggerimenti sul codice sono attivi per impostazione predefinita. Quando si impostano le preferenze, è possibile disattivare i suggerimenti sul codice o determinarne la velocità di visualizzazione. Quando i suggerimenti sul codice sono disattivati nelle preferenze, è comunque possibile visualizzare un suggerimento sul codice per un comando specifico.

Per specificare le impostazioni per i suggerimenti sul codice automatici, effettuare una delle seguenti operazioni:

- Nel pannello Azioni o nella finestra Script, selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh), fare clic su ActionScript nell'elenco Categoria, quindi abilitare o disabilitare Suggerimenti codice.
- Nel pannello Azioni, selezionare Preferenze dal menu a comparsa (nell'angolo superiore destro del pannello Azioni) e attivare o disattivare Suggerimenti codice nelle preferenze di ActionScript.

Se si attivano i suggerimenti sul codice, è possibile inoltre specificare un ritardo in secondi per la visualizzazione dei suggerimenti. Ad esempio, se si utilizza ActionScript per la prima volta, si può scegliere di visualizzare immediatamente il suggerimento sul codice. Tuttavia, se si conosce il codice da digitare e si desidera che i suggerimenti vengano visualizzati solo quando si utilizzano elementi del linguaggio poco familiari, è possibile specificare un ritardo in modo che i suggerimenti non appaiano quando non si desidera utilizzarli.

Per specificare un ritardo per i suggerimenti sul codice:

1. Nel pannello Azioni o nella finestra Script, selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh) dal menu principale.
2. Fare clic su ActionScript nell'elenco Categoria,
3. quindi utilizzare il cursore per selezionare un valore per il ritardo.
Il valore di ritardo è espresso in secondi.

Per utilizzare i suggerimenti sul codice sotto forma di descrizioni comandi:

1. Visualizzare i suggerimenti sul codice digitando una parentesi tonda aperta [()] dopo un elemento che richiede parentesi (ad esempio il nome di un metodo, un comando if o do while e così via).

Viene visualizzato il suggerimento sul codice.

The screenshot shows a code editor with the following code:
`if (`
A dropdown menu is open over the closing parenthesis, displaying the suggestion:
`1 di 2 if(condizione) {`
Below this, another code snippet is shown:
`my_array.splice(`
A dropdown menu is open over the opening parenthesis, displaying the suggestion:
`Array.splice(indice, num, elemento1, ..., elementoN)`

NOTA

Se il suggerimento sul codice non viene visualizzato, controllare di avere abilitato Suggerimenti codice nelle preferenze di ActionScript (Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh), quindi selezionare ActionScript nell'elenco Categoria). Se si desidera visualizzare i suggerimenti sul codice per una variabile o un oggetto creato, verificare di aver assegnato un nome corretto alla variabile o all'oggetto (vedere “[Informazioni sull'uso dei suffissi per attivare i suggerimenti sul codice](#)” a pagina 52) oppure di utilizzare la tipizzazione forte per la variabile o l'oggetto (vedere “[Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice](#)” a pagina 51).

2. Immettere un valore per il parametro.

Se vi sono più parametri, separare i valori con una virgola. Per funzioni o istruzioni, quali il ciclo for, separare i parametri con punto e virgola.

Con comandi di overload (funzioni o metodi che possono essere richiamati con diversi set di parametri), quali gotoAndPlay() o for, viene visualizzato un indicatore che consente di selezionare il parametro da impostare. Fare clic sui pulsanti freccia o premere Ctrl+freccia Sinistra e Ctrl+freccia Destra per selezionare il parametro.

The screenshot shows a code editor with the following code:
`for (`
A dropdown menu is open over the opening parenthesis, displaying the suggestion:
`1 di 2 for (iniziale; condizione; successivo) {`

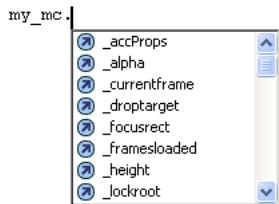
3. Per chiudere il suggerimento sul codice, effettuare una delle seguenti operazioni:

- Digitare una parentesi tonda chiusa ()].
- Fare clic al di fuori dell'istruzione.
- Premere il tasto Esc.

Per utilizzare i suggerimenti sul codice sotto forma di menu:

1. Visualizzare il suggerimento sul codice digitando un punto dopo il nome della variabile o dell'oggetto.

Viene visualizzato il menu dei suggerimenti sul codice.



NOTA

Se il suggerimento sul codice non viene visualizzato, controllare di avere abilitato Suggerimenti codice nelle preferenze di ActionScript (Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh), quindi selezionare ActionScript nell'elenco Categoria). Se si desidera visualizzare i suggerimenti sul codice per una variabile o un oggetto creato, verificare di aver assegnato un nome corretto alla variabile o all'oggetto (vedere [“Informazioni sull'uso dei suffissi per attivare i suggerimenti sul codice” a pagina 52](#)) oppure di utilizzare la tipizzazione forte per la variabile o l'oggetto (vedere [“Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice” a pagina 51](#)).

2. Per spostarsi tra i suggerimenti sul codice, usare i tasti Freccia su e Freccia giù.

3. Per selezionare una voce di menu, premere il tasto Invio o Tab oppure fare doppio clic sull'elemento.

4. Per chiudere il suggerimento sul codice, effettuare una delle seguenti operazioni:

- Selezionare una delle voci di menu.
- Fare clic sopra o sotto la finestra.
- Digitare una parentesi tonda chiusa ()] se in precedenza era stata digitata una parentesi tonda aperta [().
- Premere il tasto Esc.

Per visualizzare manualmente un suggerimento sul codice:

1. Fare clic all'interno del codice in una posizione per cui è possibile visualizzare i suggerimenti, ad esempio:
 - Dopo il punto (.) che segue un'istruzione o un comando dove è necessario immettere una proprietà o un metodo
 - Tra le parentesi [0] nel nome di un metodo
2. Effettuare una delle seguenti operazioni:■ Fare clic su Mostra suggerimento sul codice nella barra degli strumenti del pannello Azioni o della finestra Script.
- Premere Ctrl+Barra spaziatrice (Windows) o Comando+Barra spaziatrice (Macintosh).
- Quando si lavora nel pannello Azioni, selezionare Mostra suggerimento sul codice dal menu a comparsa.

Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice

Se si utilizza ActionScript 2.0, è possibile fare ricorso alla tipizzazione forte dei dati per una variabile basata su una classe incorporata, ad esempio Button, Array e così via. In questo caso, il riquadro Script visualizza i suggerimenti sul codice per la variabile. Ad esempio, si supponga di digitare il seguente codice:

```
var names:Array = new Array();  
names.
```

Non appena viene digitato il punto (.), Flash visualizza un elenco di metodi e proprietà disponibili per gli oggetti Array in un menu a comparsa, in quanto la variabile è stata digitata come un array. Per ulteriori informazioni sulla tipizzazione dei dati, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344. Per informazioni su come utilizzare i suggerimenti sul codice quando vengono visualizzati, vedere “[Uso dei suggerimenti sul codice](#)” a pagina 48.

Informazioni sull'uso dei suffissi per attivare i suggerimenti sul codice

Se si utilizza ActionScript 1 o si desidera visualizzare i suggerimenti sul codice per gli oggetti creati senza tipizzazione forte (vedere “[Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice](#)” a pagina 51), è necessario aggiungere un suffisso speciale al nome di ogni oggetto al momento della sua creazione. Ad esempio, i suffissi che attivano il suggerimento del codice per la classe Array e la classe Camera sono `_array` e `_cam`, rispettivamente. Osservare il seguente esempio di codice

```
var my_array = new Array();  
var my_cam = Camera.get();
```

è possibile digitare il codice seguente (il nome della variabile seguito da un punto):

```
my_array.  
my_cam.
```

Verranno visualizzati suggerimenti sul codice per gli oggetti Array e Camera.

Per gli oggetti che vengono visualizzati sullo stage, utilizzare il suffisso nella casella di testo Nome istanza della finestra di ispezione Proprietà. Ad esempio, per visualizzare i suggerimenti sul codice per gli oggetti MovieClip, utilizzare la finestra di ispezione Proprietà per assegnare i nomi di istanza con il suffisso `_mc` a tutti gli oggetti MovieClip. Quindi, ogni volta che si digita il nome di istanza seguito da un punto, vengono visualizzati i suggerimenti sul codice.

Sebbene i suffissi non siano obbligatori per l'attivazione dei suggerimenti sul codice se si utilizza la tipizzazione forte dei dati per un oggetto, il loro utilizzo regolare facilita la comprensione del codice.

La tabella seguente elenca i suffissi richiesti per supportare i suggerimenti automatici sul codice.

Tipo di oggetto	Suffisso variabile
Array	<code>_array</code>
Button	<code>_btn</code>
Camera	<code>_cam</code>
Color	<code>_color</code>
ContextMenu	<code>_cm</code>
ContextMenuItem	<code>_cmi</code>
Date	<code>_date</code>
Error	<code>_err</code>
LoadVars	<code>_lv</code>

Tipo di oggetto	Suffisso variabile
LocalConnection	_lc
Microphone	_mic
MovieClip	_mc
MovieClipLoader	_mcl
PrintJob	_pj
NetConnection	_nc
NetStream	_ns
SharedObject	_so
Sound	_sound
String	_str
TextField	_txt
TextFormat	_fmt
Video	_video
XML	_xml
XMLNode	_xmlnode
XMLSocket	_xmlsocket

Per informazioni su come utilizzare i suggerimenti sul codice quando vengono visualizzati, vedere “[Uso dei suggerimenti sul codice](#)” a pagina 48.

Informazioni sull'uso dei commenti per attivare i suggerimenti sul codice

È inoltre possibile usare i commenti di ActionScript per specificare la classe di un oggetto per i suggerimenti sul codice. L'esempio seguente indica ad ActionScript che la classe dell'istanza `theObject` è `Object` e così via. Se si digita `mc` seguito da un punto dopo i commenti, vengono visualizzati suggerimenti sul codice contenenti un elenco dei metodi e delle proprietà di

`MovieClip`; se si digita `theArray` seguito da un punto, viene visualizzato un menu contenente un elenco di metodi e proprietà di `Array` e così via.

```
// Object theObject;
// Array theArray;
// MovieClip theMc;
```

Macromedia consiglia tuttavia l'uso della tipizzazione forte dei dati (vedere “[Informazioni sulla tipizzazione dei dati degli oggetti per attivare i suggerimenti sul codice](#)” a pagina 51) oppure dei suffissi (vedere “[Informazioni sull'uso dei suffissi per attivare i suggerimenti sul codice](#)” a pagina 52), perché queste due tecniche attivano automaticamente i suggerimenti, rendendo il codice più comprensibile. Per ulteriori informazioni sull'uso dei suggerimenti sul codice, vedere “[Uso dei suggerimenti sul codice](#)” a pagina 48.

Formattazione di codice

È possibile specificare le impostazioni per determinare se la formattazione e il rientro del codice avvengono automaticamente o manualmente, nonché se utilizzare la mappatura dei caratteri dinamica che garantisce l'uso dei caratteri corretti in caso di lavoro con testo in più lingue.

Per impostare le opzioni di formattazione:

1. Nel pannello Azioni, selezionare Preferenze dal menu a comparsa (nell'angolo superiore destro del pannello). Nella finestra di dialogo Preferenze, selezionare Formattazione automatica.

In alternativa è possibile selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh) nella finestra Script. Nella finestra di dialogo Preferenze, selezionare Formattazione automatica.

2. Selezionare le opzioni di formattazione automatica desiderate.

Per visualizzare il risultato delle selezioni effettuate, vedere il riquadro di anteprima.

Le impostazioni eseguite nella finestra Opzioni formattazione automatica vengono applicate automaticamente al codice scritto dall'utente, ma non al codice esistente, a cui occorre applicare le impostazioni manualmente. Il codice formattato in precedenza con impostazioni diverse, importato da un editor diverso e così via deve essere formattato manualmente.

Per formattare il codice in base alle impostazioni di Formattazione automatica, effettuare una delle seguenti operazioni:

- Fare clic sul pulsante Formattazione automatica nella barra degli strumenti del pannello Azioni o della finestra Script.
- Nel pannello Azioni, selezionare Formattazione automatica dal menu a comparsa.
- Premere Ctrl+Maiusc+F (Windows) o Comando+Maiusc+F (Macintosh).
- Nella finestra Script, selezionare Strumenti > Formattazione automatica.

Per utilizzare la mappatura dei caratteri dinamica:

- Per attivare o disattivare la mappatura dei caratteri dinamica, selezionare o deselectare nella finestra di dialogo Preferenze.
- Per impostazione predefinita, la mappatura dei caratteri dinamica è disattivata, in quanto aumenta il tempo richiesto per la creazione dello script. Se si lavora con testo in più lingue, attivare la mappatura dei caratteri dinamica per garantire l'uso dei caratteri corretti.

Per usare il rientro automatico:

- Per attivare o disattivare il rientro automatico, selezionare o deselectare Rientro automatico nella finestra di dialogo Preferenze.
- Quando il rientro automatico è attivo, al testo digitato dopo un segno di parentesi aperta [(o {} viene automaticamente applicato un rientro in base alle impostazioni definite in Dimensione tabulazioni nelle preferenze di ActionScript.
- Negli script è possibile applicare un rientro a una riga selezionandola e premendo il tasto Tab. Per rimuovere il rientro, selezionare la riga e premere il Maiusc+Tab.

Uso dell'evidenziazione della sintassi

In ActionScript, come in tutti i linguaggi, la *sintassi* identifica il modo in cui gli elementi vengono combinati tra loro per produrre un significato. Se la sintassi di ActionScript è scorretta, lo script non funziona.

Quando si creano script in Flash Basic 8 e Flash Professional 8, i comandi che non sono supportati dalla versione del lettore che si desidera impostare come destinazione sono visualizzati in giallo nella casella degli strumenti Azioni. Ad esempio, se la versione SWF di Flash Player è impostata su Flash 7, gli elementi di ActionScript che sono supportati solo in Flash Player 8 vengono visualizzati in giallo nella casella degli strumenti Azioni. Per ulteriori informazioni sull'impostazione della versione del file SWF di Flash Player, vedere [Capitolo 17, “Impostazione delle opzioni di pubblicazione per il formato file SWF di Flash”](#) nella guida *Uso di Flash*.

È inoltre possibile impostare una preferenza che consente di colorare gli elementi del codice degli script di Flash man mano che vengono scritti, per evidenziare gli errori di digitazione. È possibile, ad esempio, impostare la preferenza di colorazione della sintassi in modo che le parole chiave vengano visualizzate in blu scuro. Durante la scrittura del codice, se si digita var, la parola var viene visualizzata in blu. Se invece per errore si digita vae, la parola vae resta in nero, consentendo di riconoscere immediatamente l'errore di digitazione. Per informazioni sulle parole chiave, vedere [“Informazioni sulle parole chiave” a pagina 103](#).

Per impostare le preferenze per la colorazione della sintassi durante la digitazione, effettuare una delle seguenti operazioni:

- Selezionare Modifica > Preferenze (Windows) oppure Flash > Preferenze (Macintosh) e fare clic su ActionScript nell'elenco Categoria, quindi specificare le impostazioni Colori sintassi.
-  ■ Nel pannello Azioni, selezionare Preferenze dal menu a comparsa (nell'angolo superiore destro del pannello), quindi specificare le impostazioni di Colori sintassi nelle preferenze di ActionScript.
- Con il puntatore nel riquadro Script premere Ctrl+U in Windows o Comando+U in Macintosh.

È possibile modificare le impostazioni dei colori per le parole chiave, i commenti, gli identificatori e le stringhe. Per informazioni sugli identificatori e le stringhe, vedere “[Terminologia](#)” a pagina 869 e “[Tipo di dati String](#)” a pagina 342. Per informazioni sui commenti, vedere “[Informazioni sui commenti](#)” a pagina 96.

Uso di numeri di riga e di ritorno a capo automatico

È possibile scegliere se visualizzare i numeri di riga e se mandare a capo automaticamente le righe di codice lunghe. In genere numeri di riga e ritorno a capo automatico vengono abilitati, per semplificare la modifica del codice. Con i numeri di riga è più facile scorrere nel codice e analizzarlo nella fase di modifica. L'impostazione del ritorno a capo automatico evita la necessità di scorrere orizzontalmente in righe di codice lunghe, in particolare quando si lavora nell'ambiente di creazione o con basse risoluzioni di schermo.

Per abilitare o disabilitare i numeri di riga, eseguire una delle operazioni seguenti:

-  ■ Nel pannello Azioni, selezionare Numeri di riga dal menu a comparsa.
- Nella finestra Script, selezionare Strumenti > Numeri di riga.
- Premere Ctrl+Maiusc+L (Windows) oppure Comando+Maiusc+L (Macintosh).

Per abilitare o disabilitare il ritorno a capo automatico, eseguire una delle operazioni seguenti:

-  ■ Nel pannello Azioni, selezionare A capo automatico dal menu a comparsa.
- Nella finestra Script, selezionare Strumenti > A capo automatico.
- Premere Ctrl+Maiusc+B (Windows) oppure Comando+Maiusc+B (Macintosh).

Uso dei tasti di scelta rapida Esc

È possibile aggiungere molti elementi a uno script utilizzando i tasti di scelta rapida Esc, vale a dire premendo il tasto Esc e due ulteriori tasti.

NOTA

Queste scelte rapide sono diverse rispetto ai tasti di scelta rapida che consentono di eseguire determinati comandi di menu.

Ad esempio, quando si lavora nel riquadro Script e si preme Esc+d+o, nello script viene inserito il seguente codice:

```
do {  
} while ();
```

Il punto di inserimento si trova immediatamente dopo la parola `while`, quindi è possibile cominciare a digitare la condizione. Analogamente, se si preme Esc+c+h, il seguente codice viene inserito nello script con il punto di inserimento tra le parentesi tonde `()`, affinché sia possibile iniziare a digitare subito la condizione:

```
catch () {  
}
```

Per conoscere i comandi a cui sono associati i tasti di scelta rapida Esc, vedere la casella degli strumenti di ActionScript, dove a ogni elemento è associato il relativo tasto di scelta rapida.



Per visualizzare o nascondere i tasti di scelta rapida Esc:



- Selezionare o deselectare Tasti di scelta rapida Esc dal menu a comparsa del pannello Azioni.

I tasti di scelta rapida Esc vengono visualizzati accanto agli elementi nella casella degli strumenti di ActionScript.

Visualizzazione dei caratteri nascosti

Durante la scrittura e la formattazione del codice ActionScript, si immettono spazi, caratteri di tabulazione e interruzioni di riga negli script. Tutti questi caratteri sono necessari per la struttura visiva del codice; tuttavia, il compilatore di Flash genera degli errori se incontra spazi a doppio byte che non fanno parte di un valore stringa. La visualizzazione dei caratteri nascosti nel riquadro Script consente di vedere e, quindi, rimuovere gli spazi a doppio byte.

I simboli seguenti sono usati per visualizzare i singoli caratteri nascosti:

spazio a byte singolo .

spazio a doppio byte |

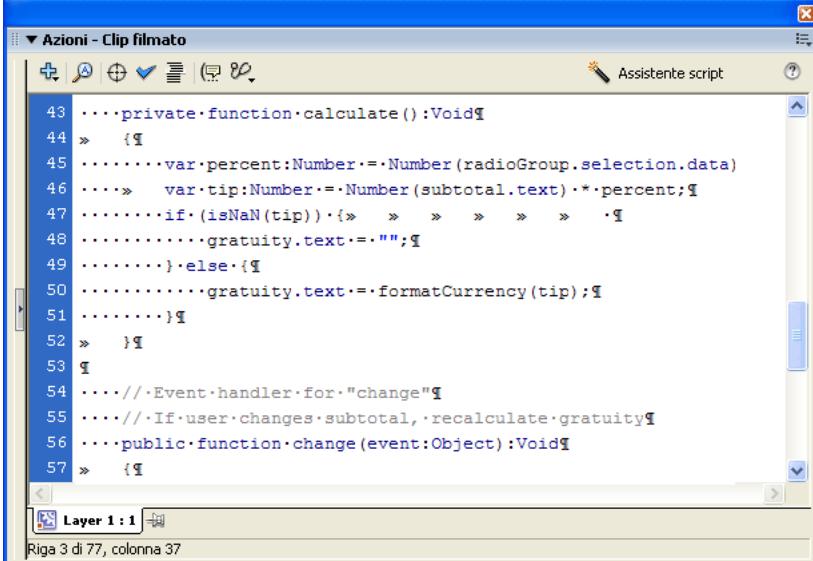
carattere di tabulazione >>

interruzione di riga ¶

Per visualizzare i caratteri nascosti, effettuare una delle seguenti operazioni:

- Selezionare Caratteri nascosti dal menu a comparsa.
- Premere Ctrl+Maiusc+8 (Windows) oppure Comando+Maiusc+8 (Macintosh).

Quando i caratteri nascosti sono visualizzati, il riquadro Script ha questo aspetto:



The screenshot shows the 'Actions - Clip filmato' panel in the Flash IDE. The code area displays several hidden characters (spaces, tabs, and line breaks) as visible symbols. The code itself is a series of ActionScript statements, including variable declarations, assignments, and function definitions. The status bar at the bottom indicates 'Layer 1 : 1' and 'Riga 3 di 77, colonna 37'.

```
43     ....private.function.calculate():Void{  
44       ¶  
45       ....var.percent:Number.=Number(radioGroup.selection.data)  
46       ....>> var.tip:Number.=Number(subtotal.text)*.percent;¶  
47       ....if.(isNaN(tip)).{>> >> >> >> >> ¶  
48       ....gratuity.text.="";¶  
49       ....}....else.{¶  
50       ....gratuity.text.=formatCurrency(tip);¶  
51       ....}¶  
52     }¶  
53 ¶  
54     ....//.Event.handler.for."change"¶  
55     ....//.If.user.changes.subtotal,.recalculate.gratuity¶  
56     ....public.function.change(event:Object):Void¶  
57     {¶
```

Uso dello strumento Trova

Lo strumento Trova consente di cercare e, eventualmente, sostituire le stringhe di testo negli script. È possibile sostituire la prima o tutte le occorrenze del testo nello script e scegliere di eseguire una ricerca di testo a lettere maiuscole o minuscole.

Per trovare del testo in uno script:

1. Nella barra degli strumenti del pannello Azioni o della finestra Script, selezionare lo strumento Trova oppure premere Ctrl+F (Windows) o Comando+F (Macintosh).
2. Immettere la stringa che si desidera individuare nello script.
3. Fare clic su Trova successivo.

Se il testo o i caratteri cercati sono presenti nel testo, le parole o i caratteri verranno evidenziati nel riquadro Script.

Per trovare e sostituire del testo in uno script:

1. Nella barra degli strumenti del pannello Azioni o della finestra Script, selezionare lo strumento Trova oppure premere Ctrl+F (Windows) o Comando+F (Macintosh).
2. Immettere la stringa che si desidera individuare e sostituire nello script.
3. Nella casella di testo Sostitisci, immettere la nuova stringa.
4. Fare clic su Trova successivo.
Se la stringa è presente nello script, viene evidenziata.
5. Per sostituire la stringa fare clic su Sostitisci oppure, per sostituire tutte le occorrenze della stringa, fare clic su Sostitisci tutto.

Dopo aver immesso una stringa di ricerca nello strumento Trova, è possibile ripetere la ricerca selezionando Trova di nuovo nel menu a comparsa.

Controllo della sintassi e della punteggiatura

Per determinare se il codice funziona come previsto, è necessario pubblicare o provare il file. Tuttavia, è anche possibile eseguire un controllo rapido del codice ActionScript senza uscire dal file FLA. Gli errori di sintassi vengono elencati nel pannello Output. È possibile inoltre verificare se sono state inserite entrambe le parentesi tonde, graffe o quadre che racchiudono un blocco di codice.

Quando si controlla la sintassi, viene eseguita la verifica dello script corrente. Se da questo script vengono chiamate classi ActionScript 2.0, le classi vengono compilate e viene verificata anche la loro sintassi. Eventuali altri script presenti nel file FLA non vengono verificati.

Per controllare la sintassi, effettuare una delle seguenti operazioni:

- Fare clic su Controlla sintassi nella barra degli strumenti del pannello Azioni o della finestra Script.
- Nel pannello Azioni, selezionare Controlla sintassi dal menu a comparsa.
- Selezionare il riquadro Script, in modo che sia attivato, quindi premere Ctrl+T (Windows) o Comando+T (Macintosh).

**N
O
T
A**

Facendo clic su Controlla sintassi in un file di classe ActionScript 2.0 esterno, il percorso globale delle classi incide sul processo. A volte possono essere generati errori anche se il percorso globale delle classi è corretto, perché il compilatore non dispone di informazioni sulla classe compilata. Per ulteriori informazioni sulla compilazione di classi, vedere "[Compilazione ed esportazione di classi](#)" [a pagina 257](#).

Per controllare la correttezza della punteggiatura, effettuare una delle seguenti operazioni:

- Fare clic tra le parentesi graffe ({}), quadre ([] o tonde ()) all'interno dello script.
- Premere Ctrl+' (apostrofo) (Windows) o Comando+' (Macintosh) per evidenziare il testo tra le parentesi graffe, quadre o tonde.

L'evidenziazione consente di verificare che per la parentesi aperta sia presente la relativa parentesi chiusa.

Importazione ed esportazione degli script

È possibile sia importare uno script nel pannello Azioni o nella finestra Script che esportare gli script in file AS esterni. Entrambe queste funzioni possono essere utili per condividere il codice fra diverse applicazioni Flash e con altri team di sviluppatori.

Per importare un file AS esterno:

- Per importare uno script esterno in uno script su cui si sta lavorando nel riquadro Script, posizionare il cursore nel punto in cui si desidera posizionare la prima riga dello script esterno, quindi effettuare una delle seguenti operazioni:
 - Nel pannello Azioni, selezionare Importa script dal menu a comparsa, oppure premere Ctrl+Maiusc+I (Windows) o Comando+Maiusc+I (Macintosh).
 - Nella finestra Script, selezionare Importa script dal menu File, oppure premere Ctrl+Maiusc+I (Windows) o Comando+Maiusc+I (Macintosh).

È anche possibile esportare uno script dal pannello Azioni. Quando si usa la finestra Script, l'esportazione non è necessaria in quanto è possibile salvare il file AS.

Per esportare uno script dal pannello Azioni:

1. Selezionare lo script da esportare, quindi selezionare Esporta script dal menu a comparsa, oppure premere Ctrl+Maiusc+X (Windows) o Comando+Maiusc+X (Macintosh).
Viene visualizzata la finestra di dialogo Salva con nome.
2. Salvare il file ActionScript (AS).

Flash supporta diversi formati di codifica dei caratteri (fra cui Unicode) ed è possibile specificare quale formato usare per l'importazione e l'esportazione degli script. Per ulteriori informazioni, vedere “Importazione ed esportazione degli script” a pagina 60 e “Importazione ed esportazione delle preferenze” a pagina 61.

Supporto di Unicode per ActionScript

Flash 8 supporta la codifica di testo Unicode per ActionScript. È prevista quindi la possibilità di inserire testi in diverse lingue nei file ActionScript. È possibile, ad esempio, inserire testi in italiano, giapponese e francese nello stesso file.

ATTENZIONE

Se viene utilizzata un'applicazione non in lingua inglese in un sistema in lingua inglese, il comando Prova filmato (vedere “Esecuzione del debug degli script” a pagina 771) non verrà eseguito correttamente se in una parte del file SWF sono presenti caratteri che non è possibile rappresentare con lo schema di codifica MBCS (Multibyte Character Set). I percorsi della lingua giapponese, ad esempio, funzionano in un sistema in lingua giapponese, ma non in un sistema in lingua italiana. Tutte le aree dell'applicazione che utilizzano il lettore esterno sono soggette a questa limitazione.

Importazione ed esportazione delle preferenze

È possibile impostare le preferenze ActionScript in modo che sia specificato il tipo di codifica da utilizzare per l'importazione o l'esportazione dei file ActionScript. È possibile selezionare la codifica UTF-8 o la codifica predefinita. UTF-8 è un formato Unicode a 8 bit; la codifica predefinita è la forma di codifica supportata dalla lingua corrente utilizzata nel sistema, denominata anche *tabella codici tradizionale*.

In generale, per importare o esportare file ActionScript in formato UTF-8, utilizzare la preferenza UTF-8. Se si importano o esportano file nella tabella codici tradizionale del sistema, utilizzare la preferenza Codifica predefinita.

Se il testo degli script non ha l'aspetto desiderato quando si apre o si importa un file, modificare l'impostazione relativa alla codifica per le importazioni. Se durante l'esportazione di file ActionScript si riceve un avviso, è possibile modificare l'impostazione relativa alla codifica per le esportazioni o disattivare l'avviso nelle preferenze di ActionScript.

Per selezionare le opzioni di codifica di testo per importare o esportare i file di ActionScript:

1. Nella finestra di dialogo Preferenze (Modifica > Preferenze (Windows) oppure Flash > Preferenze (Macintosh)) selezionare ActionScript nell'elenco Categoria.
2. In Opzioni di modifica, effettuare una delle operazioni seguenti o entrambe:
 - In Apri/Importa, selezionare Codifica UTF-8 per aprire o importare i file utilizzando la codifica Unicode, oppure selezionare Codifica predefinita per aprire o importare i file utilizzando la codifica della lingua corrente del sistema.
 - In Salva/Esporta, selezionare Codifica UTF-8 per salvare o esportare i file utilizzando la codifica Unicode oppure selezionare Codifica predefinita per salvare o esportare i file utilizzando la codifica della lingua corrente del sistema.

Per attivare o disattivare l'avviso per la codifica di esportazione:

1. Nel sistema di menu di Flash selezionare Modifica > Preferenze (Windows) oppure Flash > Preferenze (Macintosh) e fare clic su Avvertenze nel menu Categoria.
2. Selezionare o deselectrare Avverti dei conflitti di codifica nell'esportazione dei file ActionScript.

Informazioni sulle funzioni del pannello Azioni

Le seguenti funzioni sono disponibili solo nel pannello Azioni; non sono disponibili nella finestra Script. Sebbene il pannello Azioni abbia tutte le funzioni della finestra Script, quest'ultima viene utilizzata per funzionalità diverse. Il pannello Azioni deve supportare alcune funzionalità relative ai file FLA, trattate nelle sezioni successive. Per informazioni sulle funzioni disponibili sia nella finestra Script che nel pannello Azioni, vedere “[Informazioni sulla creazione di codice nel pannello Azioni e nella finestra Script](#)” a pagina 40.

Per informazioni sulle funzioni disponibili solo nel pannello Azioni, vedere le seguenti sezioni:

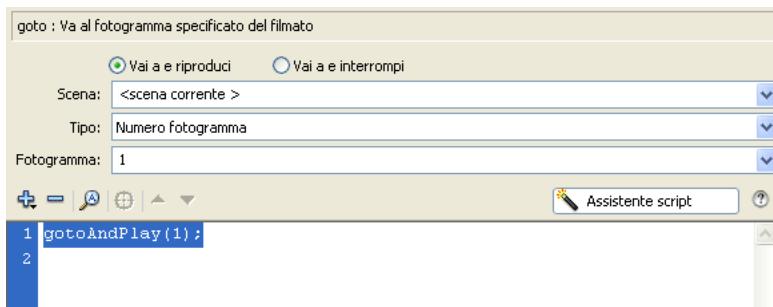
- “[Informazioni su Assistente script](#)” a pagina 63
- “[Blocco degli script nel pannello Azioni](#)” a pagina 63
- “[Inserimento dei percorsi target](#)” a pagina 65

Informazioni su Assistente script

Assistente script chiede all'utente di immettere i vari elementi necessari allo script, facilitando l'aggiunta di interattività semplice ai file SWF o alle applicazioni Flash, ed è particolarmente indicato per gli utenti che non hanno familiarità con la scrittura degli script o, comunque, che desiderano approfittare dei vantaggi offerti da questo strumento.

Usato assieme al pannello Azioni, Assistente script richiede la selezione delle opzioni e l'immissione dei parametri; ad esempio, invece di creare un nuovo script, è possibile selezionare un elemento del linguaggio dalla casella degli strumenti Azioni, o mediante il comando Aggiungi (+) nella barra degli strumenti, trascinarlo nel riquadro Script e servirsi di Assistente script per completare la scrittura.

Nell'esempio seguente, la funzione `gotoAndPlay` è stata aggiunta al riquadro Script. Assistente script visualizza tutti i prompt necessari a usare questa funzione di ActionScript, vale a dire il nome della scena, il tipo e il numero di fotogramma.



Blocco degli script nel pannello Azioni

Se non si raggruppa tutto il codice in un'unica posizione all'interno di un file FLA (come descritto in “[Organizzazione del codice ActionScript](#)” a pagina 34), o se si utilizzano i comportamenti (vedere “[Informazioni sui comportamenti](#)” a pagina 66) è possibile *bloccare* più script nel pannello Azioni per spostarsi da uno script all'altro in modo più semplice. Bloccare uno script significa avere la possibilità di mantenere la posizione del codice aperto nel pannello Azioni e fare clic tra i diversi script aperti.

Nella figura seguente, lo script associato alla posizione corrente nella linea temporale si trova nel fotogramma 1 del livello denominato Cleanup (la posizione corrente nella linea temporale viene indicata nella scheda a sinistra). Lo script è inoltre bloccato (come indicato nella scheda a destra). Sono bloccati altri due script: uno nel fotogramma 1 e l'altro nel fotogramma 15 del livello denominato Intro. È possibile spostarsi tra gli script bloccati facendo clic sulle relative schede oppure utilizzando i tasti di scelta rapida, ad esempio Ctrl+Maiusc+. (punto). Tali spostamenti non modificano la posizione corrente nella linea temporale. Come indicato nella figura seguente, nel pannello Azioni sono aperti script multipli, per spostarsi tra gli script è possibile fare clic sulle diverse schede.



SUGGERIMENTO Se il contenuto visualizzato nel riquadro Script non cambia per riflettere la posizione selezionata sulla linea temporale, è possibile che nel riquadro Script sia visualizzato uno script bloccato. Fare clic sulla prima scheda a sinistra nell'area inferiore sinistra del riquadro Script per visualizzare il codice ActionScript associato alla posizione corrente della linea temporale.

Per bloccare uno script:

1. Posizionare il puntatore nella linea temporale in modo che lo script venga visualizzato in una delle schede in basso a sinistra del riquadro Script nel pannello Azioni.
2. Effettuare una delle seguenti operazioni:
 - Fare clic sull'icona raffigurante una puntina da disegno a destra della scheda.
 - Fare clic con il pulsante destro del mouse (Windows) o premere Ctrl mentre si fa clic con il mouse (Macintosh) sulla scheda e selezionare Blocca lo script.
 - Selezionare Blocca lo script dal menu a comparsa (nella parte superiore destra del pannello Azioni).
 - Con il puntatore nel pannello Script premere Ctrl+= (segno di uguale) in Windows o Comando+= in Macintosh.

Per sbloccare uno o più script, effettuare una delle seguenti operazioni:

- Se uno script bloccato è presente in una scheda in basso a sinistra del riquadro Script nel pannello Azioni, fare clic sull'icona raffigurante la puntina da disegno a destra della scheda.
- Fare clic con il pulsante destro del mouse (Windows) o premere Ctrl mentre si fa clic con il mouse (Macintosh) su una scheda, quindi selezionare Chiudi script o Chiudi tutti gli script.

- Selezionare Chiudi script o Chiudi tutti gli script dal menu a comparsa (nella parte superiore destra del pannello Azioni).
- Con il puntatore nel riquadro Script premere Ctrl+- (segno meno) in Windows o Comando+- in Macintosh.

Per utilizzare i tasti di scelta rapida con gli script bloccati:

- Con gli script bloccati è possibile utilizzare i seguenti tasti di scelta rapida:

Azione	Tasto di scelta rapida Windows	Tasto di scelta rapida Macintosh
Blocca lo script	Ctrl+= (segno uguale)	Comando+=
Sblocca lo script	Ctrl+- (segno meno)	Comando+-
Attiva la scheda a destra	Ctrl+Maiusc+. (punto)	Comando+Maiusc+.
Attiva la scheda a sinistra	Ctrl+Maiusc+, (virgola)	Comando+Maiusc+,
Sblocca tutti gli script	Ctrl+Maiusc+- (segno meno)	Comando+Maiusc+-

Inserimento dei percorsi target

Molte delle azioni create nello script incidono sui clip filmato, i pulsanti e altre istanze di simboli. Per applicare le azioni alle istanze in una linea temporale, occorre impostare un *percorso target*, vale a dire l'indirizzo dell'istanza che si vuole usare come destinazione. È possibile impostare un percorso assoluto o un percorso relativo.

Lo strumento Percorso target, disponibile nel pannello Azioni, chiede all'utente di immettere il percorso target dell'azione selezionata nello script.

Per inserire un percorso target:

1. Selezionare e posizionare il puntatore in un'azione dello script.
 2. Fare clic su Inserisce un percorso target nella barra degli strumenti del pannello Azioni. Viene visualizzata la finestra di dialogo Inserisci percorso target.
 3. Effettuare una delle seguenti operazioni:
 - Immettere manualmente il percorso dell'istanza target.
 - Selezionare il target dall'elenco dei target disponibili.
 4. Selezionare Assoluto o Relativo come opzione del percorso target.
 5. Fare clic su OK.
- Il percorso viene aggiunto all'azione.

Informazioni sui comportamenti

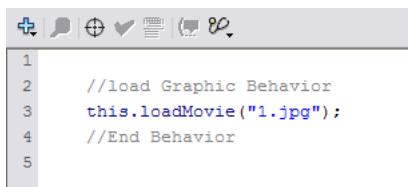
I comportamenti sono funzioni predefinite di ActionScript che è possibile associare agli oggetti del proprio documento Flash senza che sia necessario creare il codice ActionScript; forniscono funzionalità di ActionScript predefinite, quali la navigazione tra i fotogrammi, il caricamento di file SWF e JPEG esterni, il controllo dell'ordine di impilamento dei clip filmato e il trascinamento dei clip filmato.

I comportamenti possono essere usati durante la creazione di un'applicazione Flash, in modo che non sia necessario scrivere il codice ActionScript, oppure per comprendere meglio il funzionamento di ActionScript in determinate situazioni.

I comportamenti sono disponibili solo quando si lavora con documenti Flash, non con file di script esterni. Solitamente, si seleziona un oggetto di attivazione nel proprio documento, ad esempio un clip filmato o un pulsante, quindi si seleziona il pulsante Aggiungi del pannello dei Comportamenti per visualizzare i comportamenti disponibili e si seleziona quello desiderato, come descritto nell'esempio seguente:



Il comportamento viene aggiunto all'oggetto e visualizzato nel pannello Azioni.



Informazioni sulle impostazioni di pubblicazione ActionScript

È possibile modificare ActionScript in due modi: modificare ActionScript incorporato in un documento Flash con l'utilizzo del pannello Azioni, oppure modificare ActionScript in un file di script separato, esterno al documento Flash, con l'utilizzo della finestra Script. Poiché il pannello Azioni e la finestra Script sono essenzialmente due visioni diverse dello stesso Editor di ActionScript, le impostazioni e le preferenze di ActionScript in Flash vengono applicate a entrambi.

Le impostazioni di pubblicazione del documento Flash vengono modificate per cambiare la versione di ActionScript che verrà utilizzata al momento della pubblicazione del documento. È anche possibile impostare il percorso di classe del documento corrente, passando il percorso di classe ActionScript globale.

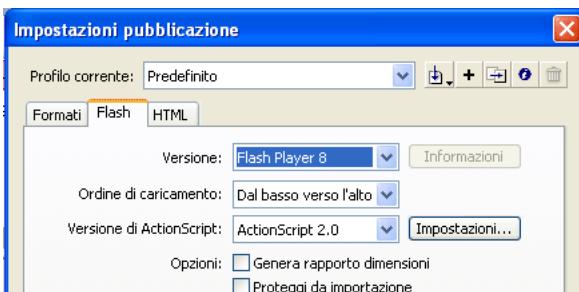
Per ulteriori informazioni sulla modifica delle impostazioni di pubblicazione ActionScript, vedere “[Modifica delle impostazioni di pubblicazione ActionScript](#)” a pagina 67. Per ulteriori informazioni sull'impostazione di un percorso di classe globale o a livello di documento, vedere “[Modifica del percorso di classe](#)” a pagina 68.

Modifica delle impostazioni di pubblicazione ActionScript

Quando si pubblica un documento Flash, per impostazione predefinita la versione di ActionScript è impostata su 2,0 e il percorso di classe viene ereditato dall'impostazione del percorso di classe globale. Se è necessario cambiare la versione o specificare un percorso di classe a livello di documento, è possibile modificare le impostazioni di pubblicazione.

Per cambiare la versione di ActionScript:

1. Selezionare File > Impostazioni pubblicazione e fare clic sulla scheda Flash.



2. Selezionare la versione ActionScript dal menu a comparsa.

Per impostazione predefinita è selezionata la versione ActionScript 2.0. Se si creano gli script in ActionScript 1.0 invece che 2.0, modificare questa impostazione prima di pubblicare il documento Flash.

Il compilatore ActionScript 2.0 compila il codice ActionScript 1.0, con l'eccezione seguente: la sintassi della barra (/) usata per indicare i percorsi dei clip filmato, ad esempio `parentClip/testMC:varName= "hello world"`) genera errori di compilazione se si seleziona la versione ActionScript 2.0. Per risolvere questo problema, riscrivere il codice mediante la notazione del punto (.) invece che della barra, oppure selezionare il compilatore ActionScript 1.0.

Per modificare il percorso di classe relativo al documento si utilizza il pulsante Impostazioni (accanto al menu a comparsa per la scelta della versione di ActionScript). Per ulteriori informazioni, vedere [“Modifica del percorso di classe” a pagina 68](#).

Modifica del percorso di classe

Quando si usa ActionScript 2.0, è anche possibile impostare un percorso di classe a livello di documento. Questa funzione è utile quando si creano le proprie classi e si desidera ignorare il percorso di classe di ActionScript globale impostato nelle preferenze.

La modifica del percorso di classe nelle impostazioni di pubblicazione viene applicata solo al file Flash corrente.

È possibile modificare il percorso di classe globale utilizzando la finestra di dialogo Preferenze. Per modificare le impostazioni del percorso di classe relativo al documento, utilizzare la finestra di dialogo Impostazioni pubblicazione per il file FLA. In entrambi i casi è possibile aggiungere percorsi di directory assoluti (ad esempio, C:/my_classes) e relativi (ad esempio, ../my_classes o ".")

Per modificare il percorso di classe globale:

- 1.** Per aprire la finestra di dialogo Preferenze, selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh).
- 2.** Selezionare ActionScript nell'elenco Categoria, quindi fare clic su Impostazioni ActionScript 2.0.
- 3.** Effettuare una delle seguenti operazioni:
 - Per aggiungere una directory al percorso di classe, fare clic sul pulsante Trova il percorso, individuare la directory da aggiungere, quindi fare clic su OK.
In alternativa, fare clic sul pulsante Aggiungi nuovo percorso (+) per aggiungere una nuova riga all'elenco Percorso di classe. Fare doppio clic sulla nuova riga, digitare un percorso relativo o assoluto, quindi fare clic su OK.

- Per modificare una directory del percorso di classe, selezionare il percorso nell'elenco Percorso di classe, fare clic sul pulsante Trova il percorso, individuare la directory da aggiungere, quindi fare clic su OK.
In alternativa, fare doppio clic sul percorso nell'elenco Percorso di classe, digitare il percorso desiderato e fare clic su OK.
- Per eliminare una directory dal percorso di classe, selezionarla nell'elenco Percorso di classe e fare clic sul pulsante Elimina il percorso selezionato.

NOTA

Non eliminare il percorso di classe globale assoluto (vedere Percorsi di classe globali e del documento) perché viene utilizzato da Flash per accedere alle classi incorporate. Se il percorso è stato eliminato inavvertitamente, reinserirlo aggiungendo \$(LocalData)/Classes come nuovo percorso di classe.

Per modificare il percorso di classe relativo a un documento:

1. Selezionare File > Impostazioni pubblicazioni per aprire la finestra di dialogo Impostazioni pubblicazioni.
2. Fare clic sulla scheda Flash.
3. Fare clic su Impostazioni, in corrispondenza del menu a comparsa Versione di ActionScript.
4. Effettuare una delle seguenti operazioni:
 - Per aggiungere una directory al percorso di classe, fare clic sul pulsante Trova il percorso, individuare la directory da aggiungere, quindi fare clic su OK.
In alternativa, fare clic sul pulsante Aggiungi nuovo percorso (+) per aggiungere una nuova riga all'elenco Percorso di classe. Fare doppio clic sulla nuova riga, digitare un percorso relativo o assoluto, quindi fare clic su OK.
 - Per modificare una directory del percorso di classe, selezionare il percorso nell'elenco Percorso di classe, fare clic sul pulsante Trova il percorso, individuare la directory da aggiungere, quindi fare clic su OK.
In alternativa, fare doppio clic sul percorso nell'elenco Percorso di classe, digitare il percorso desiderato e fare clic su OK.
 - Per eliminare una directory dal percorso di classe, selezionarla nell'elenco Percorso di classe e fare clic sul pulsante Elimina il percorso selezionato.

Per ulteriori informazioni sull'impostazione e la modifica dei percorsi di classe, vedere [“Informazioni sull'impostazione e la modifica del percorso di classe” a pagina 214](#).

File di configurazione installati con Flash 8

Quando si installa Flash Basic 8 o Flash Professional 8, nel sistema vengono installati diversi file e cartelle di configurazione relativi ad ActionScript, che possono essere utilizzati per effettuare determinate configurazioni per l'ambiente di creazione. Le modifiche vanno effettuate prestando molta attenzione; realizzare sempre una copia di backup dei file che si intende modificare.

Cartella Classes di ActionScript Contiene tutte le classi ActionScript (file AS) che sono incluse in Flash Professional 8 o Flash Basic 8. I percorsi, solitamente, sono i seguenti:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\Classes.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/Classes.

La cartella Classes è organizzata in cartelle contenenti directory, che a loro volta contengono le classi per Flash Player 7 (FP7) e Flash Player 8 (FP8). Contiene inoltre una directory per il pacchetto mx (mx), utilizzato in entrambi i lettori e una per i file ASO (aso). Per ulteriori informazioni sui file ASO, vedere “[Utilizzo dei file ASO](#)” a pagina 258. Per ulteriori informazioni sull'organizzazione di questa directory, vedere il file Readme nella cartella Classes.

Cartella Include Contiene tutti i file include globali di ActionScript; il percorso è il seguente:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\Include.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/Include.

File di configurazione ActionsPanel.xml Contiene il file di configurazione per i suggerimenti sul codice ActionScript; si trova nel seguente percorso:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\ActionsPanel\ActionScript1_2.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/ActionsPanel/ActionScript_1_2.

File di configurazione AsColorSyntax.xml File di configurazione relativo ai colori del codice ActionScript; situato nella cartella:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\ActionsPanel.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/ActionsPanel.

Informazioni su ActionScript

Le funzionalità di programmazione orientate agli oggetti (OOP, Object-Oriented Programming) di ActionScript 2.0 si basano sulla proposta ECMAScript 4, attualmente in corso di sviluppo da parte di ECMA TC39-TG1 (vedere www.mozilla.org/js/language/es4/index.html). La proposta ECMA-4 non è ancora diventata standard ed è ancora in fase di modifica, pertanto ActionScript 2.0 è basato liberamente su tale specifica.

ActionScript 2.0 supporta tutti gli elementi standard del linguaggio ActionScript e consente di creare script più conformi agli standard utilizzati in altri linguaggi orientati agli oggetti, ad esempio Java. Questo linguaggio si rivolge soprattutto agli sviluppatori di Flash intermedi o avanzati che realizzano applicazioni che richiedono l'implementazione di classi e sottoclassi. Con ActionScript 2.0 è possibile dichiarare il tipo di oggetto di una variabile quando questa viene creata (vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344). Sono inoltre stati migliorati in modo significativo gli errori del compilatore (vedere [Appendice A, “Messaggi di errore”](#) a pagina 835).

Gli aspetti rilevanti di ActionScript 2.0 includono i seguenti punti:

- Gli script che utilizzano il linguaggio ActionScript 2.0 per definire classi o interfacce devono essere memorizzati come file di script esterni, con una singola classe definita in ogni script. Le classi e le interfacce non possono pertanto essere definite nel pannello Azioni.
- È possibile importare singoli file di classi implicitamente, memorizzandoli in una posizione specificata da percorsi di ricerca globali o specifici del documento e includendoli quindi in uno script, oppure esplicitamente mediante il comando `import`. I pacchetti, ovvero le raccolte di file di classi presenti in una directory, possono essere importati utilizzando caratteri jolly.

- Le applicazioni sviluppate con ActionScript 2.0 sono supportate da Flash Player 6 e versioni successive.

ATTENZIONE

L'impostazione di pubblicazione predefinita per i nuovi file creati in Flash 8 è ActionScript 2.0. Se si desidera modificare un file FLA esistente con ActionScript 1.0 per utilizzare la sintassi di ActionScript 2.0, verificare che nelle impostazioni di pubblicazione del file FLA sia specificato ActionScript 2.0. In caso contrario, il file non può essere compilato correttamente nonostante Flash non generi errori del compilatore.

Per ulteriori informazioni sull'uso di ActionScript 2.0 per realizzare programmi orientati agli oggetti in Flash, consultare il [Capitolo 6, "Classi" a pagina 197](#).

Sebbene l'uso di ActionScript 2.0 sia consigliato, è possibile continuare a utilizzare la sintassi di ActionScript 1.0, in particolare se si eseguono operazioni tradizionali, come semplici animazioni che non richiedono l'interazione dell'utente.

Descrizione di ActionScript

Di seguito sono elencate le caratteristiche principali di ActionScript 2.0:

Modello di programmazione orientato agli oggetti (OOP) La caratteristica principale di ActionScript 2.0 è rappresentata dal modello per la creazione di programmi orientati agli oggetti, già noto agli utenti. ActionScript 2.0 implementa numerose parole chiave e diversi concetti orientati agli oggetti, ad esempio i concetti di *classe*, *interfaccia* e *pacchetti*, con cui gli utenti che programmano in Java hanno già dimestichezza.

Il modello OOP fornito con ActionScript 2.0 è una "formalizzazione sintattica" del metodo a catena di prototipi utilizzato nelle versioni precedenti di Macromedia Flash per creare oggetti e determinare l'ereditarietà. Con ActionScript 2.0 è possibile creare classi personalizzate ed estendere le classi Flash incorporate.

Tipizzazione forte dei dati ActionScript 2.0 consente di specificare esplicitamente i tipi di dati per le variabili, i parametri delle funzioni e i tipi restituiti dalle funzioni. Ad esempio, il codice seguente dichiara una variabile denominata `userName` di tipo Stringa (uno dei tipi di dati incorporati di ActionScript, o classi).

```
var userName:String = "";
```

Avvisi ed errori del compilatore Le due caratteristiche precedenti, ovvero il modello OOP e la tipizzazione forte dei dati, consentono allo strumento di creazione e al compilatore di visualizzare messaggi di avviso e di errore che facilitano il rilevamento di errori nell'applicazione in modo più rapido rispetto alla versione precedente di Flash.

Se si utilizza ActionScript 2.0, verificare che nelle impostazioni di pubblicazione per il file FLA sia specificato ActionScript 2.0. Questo valore è l'impostazione predefinita per i file creati in Flash MX 2004 e Flash 8. Se invece si apre un file FLA di una versione precedente che utilizza ActionScript 1,0 e si desidera convertirlo in ActionScript 2.0, modificare le impostazioni di pubblicazione in ActionScript 2.0. In caso contrario, il file FLA non viene compilato correttamente, anche se non vengono generati errori.

Informazioni sulla scelta tra ActionScript 1.0 e ActionScript 2.0

Quando si crea un nuovo documento o una nuova applicazione in Flash è necessario decidere come organizzare i file associati. In alcuni progetti è possibile l'uso delle classi, ad esempio se si creano applicazioni o file FLA complessi. Tuttavia non per tutti i documenti si utilizzano le classi. Per molti esempi di piccole dimensioni della documentazione, ad esempio, le classi non vengono utilizzate. L'uso delle classi per memorizzare le funzionalità non rappresenta la soluzione migliore né la soluzione più semplice nel caso di applicazioni di piccole dimensioni o file FLA semplici. In questi casi, l'inserimento di codice ActionScript all'interno del documento risulta spesso più efficiente. Il codice dovrebbe essere inserito totalmente nella linea temporale nel minor numero di fotogrammi possibile e non in istanze (come pulsanti o clip filmato) all'interno di un file FLA.

Quando si crea un progetto di piccole dimensioni, l'uso di classi o codice esterno per organizzare il codice ActionScript comporta in genere più lavoro rispetto all'aggiunta di codice ActionScript all'interno del file FLA. A volte è più semplice mantenere tutto il codice ActionScript all'interno del file FLA anziché inserirlo in una classe importata; questo non significa che si deve necessariamente usare ActionScript 1.0. È possibile inserire il codice all'interno del file FLA utilizzando ActionScript 2.0 con la tipizzazione forte dei dati e i relativi nuovi metodi e proprietà. La sintassi di ActionScript 2.0 è inoltre conforme agli standard di altri linguaggi di programmazione. Il linguaggio è più semplice da apprendere. Ad esempio, ActionScript risulta subito familiare se già si lavora con un altro linguaggio basato sulla stessa struttura e gli stessi standard di sintassi, oppure, è possibile sfruttare le conoscenze acquisite se si decide di apprendere altri linguaggi in futuro. Con ActionScript 2.0 è possibile adottare un approccio orientato agli oggetti per lo sviluppo di applicazioni utilizzando un insieme di elementi del linguaggio che possono risultare utili per lo sviluppo dell'applicazione. In alcuni casi non è possibile scegliere la versione di ActionScript da utilizzare. Se si crea un file SWF destinato a una versione non recente di Flash Player, ad esempio un'applicazione per dispositivi mobili, è necessario utilizzare ActionScript 1.0 per motivi di compatibilità.

Indipendentemente dalla versione di ActionScript utilizzata, è comunque necessario seguire le procedure consigliate. Infatti molte di queste procedure sono valide per entrambe le versioni, ad esempio la coerenza nell'uso delle maiuscole, l'utilizzo del completamento del codice, il miglioramento della leggibilità, l'utilizzo di parole diverse dalle parole chiave per i nomi di istanza e l'adozione di convenzioni coerenti per l'assegnazione dei nomi.

Se si prevede di aggiornare l'applicazione in versioni future di Flash o di aumentarne le dimensioni o la complessità, si consiglia di utilizzare ActionScript 2.0 e le classi in modo da facilitare l'aggiornamento e la modifica dell'applicazione.

ActionScript e Flash Player

Se si compila un file SWF contenente codice ActionScript 2.0 con le impostazioni di pubblicazione impostate su Flash Player 6 e ActionScript 1.0, il codice funziona, a condizione che non utilizzi classi ActionScript 2.0. Il codice non prevede la distinzione tra maiuscole e minuscole, Flash Player invece esegue tale distinzione. Di conseguenza, se il file SWF viene compilato con Impostazioni pubblicazione impostato su Flash Player 7 o 8 e ActionScript 1.0, Flash applica la distinzione tra maiuscole e minuscole.

Le annotazioni dei tipi di dati (tipizzazione forte dei dati) vengono applicate durante la compilazione per Flash Player 7 e 8 quando le impostazioni di pubblicazione sono impostate su ActionScript 2.0.

Quando si pubblicano le applicazioni, si compila il codice ActionScript 2.0 in base al codice byte di ActionScript 1.0, così da poter impostare Flash Player 6, 7 o 8 mentre si lavora con ActionScript 2.0.

Nozioni fondamentali sul linguaggio e la sintassi

Apprendere la sintassi e le istruzioni ActionScript è un po' come imparare a formulare le frasi mettendo insieme le parole e i paragrafi unendo le frasi. Le regole di ActionScript sono piuttosto semplici. In italiano, ad esempio, per terminare una frase si utilizza il punto; in ActionScript per terminare un'istruzione si usa il punto e virgola. Nel linguaggio ActionScript è possibile specificare un'azione `stop()` per interrompere l'indicatore di riproduzione di un'istanza di clip filmato o il ciclo di un file SWF oppure scrivere migliaia di righe di codice per un'applicazione interattiva di Internet banking. Le operazioni eseguibili possono quindi essere da molto semplici a molto complesse.

Nel [Capitolo 10, “Dati e tipi di dati”](#) è stato illustrato l'uso dei dati nel linguaggio ActionScript e le relative modalità di formattazione. Questo capitolo dimostra come costruire le istruzioni ActionScript tramite la *sintassi*. Contiene molti brevi frammenti di codice e alcuni esempi per dimostrare i concetti fondamentali del linguaggio. I prossimi capitoli contengono esempi di codice più lunghi e complessi che includono i concetti fondamentali illustrati in questo capitolo.

Le regole generali descritte in questa sezione riguardano tutto il codice ActionScript. La maggior parte dei termini ActionScript è associata a requisiti individuali; per le regole relative a un termine specifico, vedere la voce corrispondente nella *Guida di riferimento di ActionScript 2.0*.

Appicare le regole di base del linguaggio in modo tale da creare programmi accattivanti può essere una sfida per chi inizia a utilizzare ActionScript. Per ulteriori informazioni su come applicare le regole descritte in questa sezione, vedere [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0” a pagina 791](#).

NOTA

In questo capitolo si aggiunge il codice ActionScript direttamente in un fotogramma sulla linea temporale. Nei prossimi capitoli, si utilizzano invece le classi per separare il codice ActionScript dal file FLA.

Per ulteriori informazioni sull'utilizzo della sintassi ActionScript e per le nozioni fondamentali sul linguaggio, consultare i seguenti argomenti:

Informazioni sulla sintassi, le istruzioni e le espressioni	76
Informazioni sulla sintassi del punto e sui percorsi target	80
Informazioni sui segni di punteggiatura utilizzati nel linguaggio.....	88
Informazioni su costanti e parole chiave	100
Informazioni sulle istruzioni	106
Informazioni sugli array	129
Informazioni sugli operatori	143

Informazioni sulla sintassi, le istruzioni e le espressioni

Il linguaggio ActionScript è composto da classi incorporate. È necessario utilizzare una *sintassi* ActionScript corretta per costruire le istruzioni affinché il codice possa essere compilato ed eseguito correttamente in Flash. In questo caso, per sintassi si intende la grammatica e l'ortografia del linguaggio utilizzato per la programmazione. In caso di sintassi non corretta, il compilatore non è in grado di comprendere il codice, pertanto vengono visualizzati errori o avvisi nel pannello Output quando si tenta di provare il documento nell'ambiente di prova. La sintassi può quindi essere definita come un insieme di regole e linee guida che aiutano a creare codice ActionScript corretto.

Un'*istruzione* è come un comando impartito al file FLA affinché venga eseguita un'operazione, ad esempio eseguire una determinata azione. È possibile ad esempio ricorrere a un'istruzione condizionale per determinare se qualcosa è vero o esiste e quindi eseguire determinate azioni, ad esempio funzioni o espressioni, in base al fatto che la condizione sia soddisfatta o meno. L'istruzione `if` è un'istruzione condizionale e restituisce una condizione con lo scopo di determinare l'azione successiva che deve essere eseguita nel codice.

```
// Istruzione if
if (condizione) {
    // Istruzioni;
}
```

Per ulteriori informazioni sulle istruzioni, vedere “[Informazioni sulle istruzioni](#)” a pagina 106.

Espressioni: a differenza delle istruzioni, sono combinazioni valide di simboli ActionScript che rappresentano un valore. Le espressioni sono associate a *valori*, mentre i valori e le proprietà sono associati a *tipi*. Un'espressione può essere costituita da operatori e operandi, valori, funzioni e procedure e segue le regole ActionScript di precedenza e di associazione. In genere Flash Player interpreta l'espressione e quindi restituisce un valore utilizzabile nell'applicazione.

Il seguente codice, ad esempio, corrisponde a un'espressione:

x + 2

Nell'espressione precedente, x e 2 sono operandi e + è un operatore. Per ulteriori informazioni su operatori e operandi, vedere “[Informazioni sugli operatori](#)” a pagina 143. Per ulteriori informazioni su oggetti e proprietà, consultare “[Tipo di dati oggetto](#)” a pagina 341.

La modalità di formattazione del codice ActionScript ne determina inoltre la gestibilità. Sarebbe, ad esempio, molto difficile leggere la logica di un file FLA che non contiene commenti o rientri oppure per cui sono state adottate convenzioni incoerenti per la formattazione e per l'assegnazione dei nomi. Se vengono applicati rientri ai blocchi di codice ActionScript, ad esempio ai cicli e alle istruzioni `if`, il codice risulta di più facile lettura ed è più semplice eseguirne il debug in caso di problemi. Per ulteriori informazioni sulla formattazione di ActionScript, vedere “[Formattazione della sintassi ActionScript](#)” a pagina 825. In queste sezioni è anche possibile visualizzare la corretta formattazione di ActionScript.

Per ulteriori informazioni sulla sintassi e sulle nozioni fondamentali del linguaggio, consultare i seguenti argomenti:

- “[Differenze tra ActionScript e JavaScript](#)”
- “[Informazioni sulla distinzione tra maiuscole e minuscole](#)”

Differenze tra ActionScript e JavaScript

Nelle sue caratteristiche di base, ActionScript è simile al linguaggio di programmazione JavaScript. Non occorre conoscere JavaScript per utilizzare e apprendere ActionScript, tuttavia, se già si conosce questo linguaggio, ActionScript risulterà familiare.

Lo scopo del presente manuale esula dall'insegnare la programmazione in generale; a tal fine sono disponibili numerose risorse che forniscono informazioni sui concetti generali di programmazione e sul linguaggio JavaScript.

- La specifica del linguaggio ECMAScript (ECMA-262) edizione 3 è derivata da JavaScript e viene usato come standard internazionale per questo linguaggio. ActionScript si basa su questa specifica. Per ulteriori informazioni, vedere www.ecma-international.org/publications/standards/Ecma-262.htm.
- Il sito di Java Technology offre esercitazioni sulla programmazione orientata a oggetti (<http://java.sun.com/docs/books/tutorial/java/index.html>), progettate per il linguaggio Java ma utili anche per comprendere vari concetti che valgono anche per ActionScript.

Di seguito sono elencate alcune differenze tra ActionScript e JavaScript:

- ActionScript non supporta oggetti specifici per determinati browser, quali Document, Window e Anchor.
- ActionScript non supporta completamente tutti gli oggetti incorporati di JavaScript.
- ActionScript non supporta alcuni costrutti sintattici di JavaScript, ad esempio le etichette di istruzioni.
- In ActionScript la funzione `eval()` può fare riferimento solo a variabili.
- ActionScript 2.0 supporta molte funzionalità non presenti nella specifica ECMA-262, ad esempio le classi e la tipizzazione forte. Molte di queste funzionalità sono modellate dopo la specifica del linguaggio ECMAScript (ECMA-262) edizione 3 (vedere www.ecma-international.org/publications/standards/Ecma-262.htm).
- ActionScript non supporta le espressioni regolari che utilizzano l'oggetto RegExp. L'oggetto RegExp è tuttavia supportato da Macromedia Central. Per ulteriori informazioni su Macromedia Central, visitare il sito all'indirizzo www.macromedia.com/software/central.

Informazioni sulla distinzione tra maiuscole e minuscole

Quando si scrive codice ActionScript per Flash Player 7 e per le versioni successive, è necessario fare distinzione tra maiuscole e minuscole. Le variabili con maiuscole e minuscole non perfettamente uguali vengono considerate diverse, come nel codice seguente:

```
// Maiuscole e minuscole miste
var firstName:String = "Jimmy";
// Tutte minuscole
trace(firstname); // undefined
```

In alternativa, è possibile scrivere quanto segue:

```
// Nei file pubblicati per Flash Player 8
// e ActionScript 1.0 o ActionScript 2.0
//
// Imposta le proprietà di due oggetti diversi
cat.hilite = true;
CAT.hilite = true;

// Crea tre variabili diverse
var myVar:Number = 10;
var myvar:Number = 10;
var mYvAr:Number = 10;
```

NOTA

Si sconsiglia di utilizzare maiuscole e minuscole diverse per differenziare le variabili. Per ulteriori informazioni sull'assegnazione di nomi alle variabili, consultare il [Capitolo 19, "Procedure ottimali e convenzioni di codifica per ActionScript 2.0"](#) a pagina 791.

Quando si esegue la pubblicazione per Flash Player 6 e per le versioni precedenti, Flash traccia la stringa `Jimmy` nel pannello Output. Dato che Flash Player 7 e le versioni successive fanno distinzione tra maiuscole e minuscole, `firstName` e `firstname` sono considerate variabili diverse (quando si utilizza ActionScript 1.0 o ActionScript 2.0). Questo aspetto è importante e deve essere sempre tenuto presente. Se si creano file FLA per Flash Player 6 o per le versioni precedenti utilizzando variabili con maiuscole e minuscole non corrispondenti, il funzionamento potrebbe non essere assicurato dopo la conversione del file o dell'applicazione per una versione successiva di Flash Player.

Per questo motivo si consiglia di utilizzare maiuscole e minuscole in modo coerente, ad esempio in base alle convezioni illustrate nel presente manuale. In tal modo risulta anche più semplice riconoscere le variabili dalle classi e dai nomi di funzione. Non differenziare due identificatori solo in base alle maiuscole e alle minuscole, ma modificare l'istanza, la variabile o il nome di classe. Per ulteriori informazioni sulle convenzioni di codifica, consultare il [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0”](#) a pagina 791.

La distinzione tra maiuscole e minuscole può avere un impatto importante se si lavora con un servizio Web che utilizza regole proprie per assegnare i nomi alle variabili e per definire le maiuscole e minuscole delle variabili che vengono restituite al file SWF dal server. Se, ad esempio, si utilizza un servizio Web ColdFusion, i nomi delle proprietà in una struttura o un oggetto potrebbero avere tutte le lettere maiuscole, ad esempio `NOME`. Se non si utilizza la stessa convenzione in Flash, si potrebbero ottenere risultati imprevisti.



La distinzione tra maiuscole e minuscole interessa anche le variabili esterne caricate in un file SWF, ad esempio quelle caricate con `LoadVars.load()`.

La distinzione tra maiuscole e minuscole viene implementata anche per gli script esterni, quali i file di classe di ActionScript 2.0, gli script importati tramite il comando `#include` e gli script presenti in un file FLA. Se vengono visualizzati errori di runtime in caso di esportazione in più versioni di Flash Player, verificare sia i file di script esterni che gli script nei file FLA per controllare che le maiuscole e le minuscole siano state utilizzate in modo coerente.

La distinzione tra maiuscole e minuscole viene eseguita separatamente per ogni file SWF. Se un'applicazione Flash Player 8 che fa distinzione tra maiuscole e minuscole con implementazione rigorosa chiama un Flash Player file SWF versione 6 con implementazione non rigorosa, il codice ActionScript eseguito nel file SWF versione 6 non ha implementazione rigorosa. Se, ad esempio, si utilizza `loadMovie()` per caricare un file SWF di Flash Player 6 in un file SWF di Flash Player 8, per la versione 6 del file SWF non viene fatta distinzione tra maiuscole e minuscole, mentre per la versione 8 del file SWF viene fatta tale distinzione.

Quando è attivata la funzione di colorazione della sintassi, gli elementi del linguaggio scritti con il corretto uso delle maiuscole vengono visualizzati in blu per impostazione predefinita. Per ulteriori informazioni, vedere “[Informazioni sulle parole riservate](#)” a pagina 104.

Informazioni sulla sintassi del punto e sui percorsi target

In ActionScript si utilizza l'operatore punto (.) (*sintassi del punto*) per accedere alle proprietà o ai metodi che appartengono a un oggetto o a un'istanza sullo stage. L'operatore punto consente inoltre di identificare il percorso target di un'istanza (ad esempio un clip filmato), di una variabile, di una funzione o di un oggetto.

Un'espressione in cui è utilizzato questo tipo di sintassi inizia con il nome dell'oggetto o del clip filmato seguito da un punto e termina con l'elemento che si desidera specificare. La sezione che segue dimostra come scrivere espressioni con la sintassi del punto.

Per controllare un clip filmato, un file SWF caricato o un pulsante, è necessario specificare un *percorso target*. I percorsi target sono indirizzi gerarchici dei nomi di istanze dei clip filmato, delle variabili e degli oggetti di un file SWF. Per specificare il percorso target di un clip filmato o di un pulsante, è necessario assegnare al clip o al pulsante un nome di istanza. Per assegnare un nome all'istanza di un clip filmato, selezionare l'istanza e digitare il nome dell'istanza nella finestra di ispezione Proprietà. In alternativa è possibile specificare tale nome tramite codice se si crea l'istanza con ActionScript. Un percorso target consente di assegnare un'azione a un clip filmato oppure di ottenere o impostare il valore di una variabile o di una proprietà.

Per ulteriori informazioni sull'assegnazione di un nome di istanza e sull'uso della sintassi del punto per identificare un'istanza, vedere i seguenti argomenti:

- “[Informazioni sull'uso della sintassi del punto per identificare un'istanza](#)” a pagina 81.
- “[Informazioni su area di validità e identificazione](#)” a pagina 86
- “[Uso del pulsante Percorso target](#)” a pagina 87
- “[Informazioni sulla sintassi della barra](#)” a pagina 87

Per ulteriori informazioni su oggetti e proprietà, consultare “[Tipo di dati oggetto](#)” a pagina 341.

Informazioni sull'uso della sintassi del punto per identificare un'istanza

Per scrivere codice ActionScript che controlli un'istanza come un clip filmato o gestisca le risorse di un file SWF caricato, specificare il nome e l'indirizzo dell'istanza nel codice, ovvero il *percorso target*. Per identificare oggetti (o farvi riferimento) in un file SWF, si utilizza la sintassi del punto, detta anche *notazione del punto*. È necessario ad esempio identificare un'istanza di clip filmato o di un pulsante prima di applicarvi un'azione. La sintassi del punto aiuta a creare il percorso dell'istanza da identificare. Il percorso dell'istanza da identificare è a volte detto percorso target.

I file FLA sono associati a una gerarchia particolare. È possibile creare istanze sullo stage o utilizzare ActionScript. È inoltre possibile creare istanze all'interno di altre istanze o istanze nidificate all'interno di numerose altre istanze. Ogni istanza può essere gestita se le si assegna un nome.

Per assegnare un nome a un'istanza si utilizza un *nome di istanza*, specificabile nei due modi seguenti:

- Manualmente tramite selezione di un'istanza e immissione del nome di istanza nella finestra di ispezione Proprietà (quando un'istanza si trova sullo stage)
- Dinamicamente tramite ActionScript. In questo caso le istanze vengono create con ActionScript e al momento della creazione viene loro assegnato un nome.

Per assegnare un nome di istanza nella finestra di ispezione Proprietà, digitare un nome nella casella di testo Nome istanza.

È inoltre possibile assegnare un nome di istanza a un oggetto creato tramite ActionScript, come illustrato nel codice seguente:

```
this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth());  
pic_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Questo codice crea un nuovo clip filmato e gli assegna il nome di istanza pic_mc. L'istanza pic_mc può quindi essere gestita tramite codice; è possibile ad esempio caricarvi un'immagine come dimostrato nel codice riportato in precedenza.

Per ulteriori informazioni sull'area di validità, vedere “[Informazioni su area di validità e identificazione](#)” a pagina 86 e “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362.

Identificazione di un'istanza

Per eseguire operazioni in un file SWF è necessario identificare l'istanza desiderata e assegnarvi un'azione o modificare le relative proprietà. In genere è necessario definire la posizione dell'istanza nel file SWF (ad esempio in quale linea temporale si trova o all'interno di quale istanza è nidificata) creando il percorso target. In un file FLA in genere vengono assegnati molti nomi di istanza e si aggiunge codice che utilizza tali nomi di istanza, identificando l'istanza desiderata e associandovi un'azione, ad esempio lo spostamento dell'indicatore di riproduzione o l'apertura di una pagina Web. Per ulteriori informazioni su oggetti e proprietà, consultare “[Tipo di dati oggetto](#)” a pagina 341.

Per identificare un'istanza:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare File > Salva con nome e assegnare al file il nome **target.fla**.
3. Utilizzando lo strumento Ovale, disegnare una forma sullo stage, un ovale di qualsiasi dimensioni e colore.
4. Utilizzare lo strumento Selezione per selezionare l'ovale sullo stage.

SUGGERIMENTO

Non dimenticare di selezionare il tratto e il riempimento, se necessario.

5. Selezionare Elabora > Converti in simbolo, l'opzione Clip filmato, quindi fare clic su OK per creare il simbolo.
6. Selezionare il clip filmato sullo stage e assegnarvi il nome di istanza **myClip** nella finestra di ispezione Proprietà.
7. Inserire un nuovo livello e rinominarlo **actions**.
8. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
myClip._xscale = 50;
```

Questa riga di codice identifica l'istanza `myClip` sullo stage. L'istanza viene ridimensionata da ActionScript alla metà della larghezza originale. Dato che il codice ActionScript si trova sulla stessa linea temporale del simbolo del clip filmato, occorre semplicemente identificare l'istanza tramite il nome di istanza. Se l'istanza si trova su una linea temporale diversa o nidificata all'interno di un'altra istanza, modificare il percorso target di conseguenza.

Identificazione di un'istanza nidificata

È inoltre possibile identificare istanze nidificate all'interno di altre istanze, ad esempio se si desidera inserire una seconda istanza di clip filmato all'interno dell'istanza myClip dell'esercizio in “[Identificazione di un'istanza](#)” a pagina 82. L'istanza nidificata può anche essere identificata tramite ActionScript. Prima di continuare con l'esercizio seguente, completare l'esercizio in “[Identificazione di un'istanza](#)” a pagina 82 e quindi effettuare i passaggi seguenti per identificare un'istanza nidificata.

Per identificare un'istanza nidificata:

1. Aprire target.fla dalla procedura di indirizzamento di un'istanza e rinominarlo come **target2.fla**.
2. Fare doppio clic sull'istanza myClip sullo stage.
3. Selezionare lo strumento Ovale per disegnare un altro ovale all'interno dell'istanza myClip.
4. Selezionare la nuova forma, quindi Elabora > Converti in simbolo.
5. Selezionare l'opzione Clip filmato e quindi OK.
6. Nella casella di testo Nome istanza della finestra di ispezione Proprietà, la nuova istanza e digitare **myOtherClip**.
7. Fare clic su Scena 1 nella barra di modifica per tornare alla linea temporale principale.
8. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
myClip.myOtherClip._xscale = 50;
```

L'istanza myOtherClip viene ridimensionata da ActionScript al 50% della larghezza corrente. Dato che il file target.fla ha modificato la proprietà `_xscale` delle istanze myClip, e myOtherClip è un simbolo nidificato, la larghezza di myOtherClip è il 25% di quella originale.

Se si lavora con clip filmato nidificati associati a linee temporali proprie, è possibile gestire l'indicatore di riproduzione in una linea temporale di un'istanza nidificata utilizzando codice simile al frammento seguente:

```
myClip.nestedClip.gotoAndPlay(15);  
myClip.someOtherClip.gotoAndStop("tweenIn");
```

Il clip modificato (ad esempio `nestedClip`) viene riportato subito prima dell'azione. Nelle sezioni seguenti questo aspetto è trattato in modo più approfondito.

Non è solo possibile accedere a metodi e proprietà predefiniti di istanze dello stage, come dimostrato negli esempi precedenti, ma è anche possibile impostare una variabile all'interno di un clip filmato, come nel codice seguente, in cui viene impostata una variabile nel clip filmato starClip:

```
starClip.speed = 1.1;  
starClip.gravity = 0.8;
```

Se le variabili relative alla velocità o alla gravità fossero state valide in precedenza nell'istanza del clip filmato starClip, i valori precedenti sarebbero stati sovrascritti in seguito all'impostazione dei nuovi valori. Sarà possibile aggiungere nuove proprietà nel clip filmato starClip, poiché la classe MovieClip è stata definita con la `dynamic`. La parola chiave `dynamic` specifica che gli oggetti basati sulla classe specificata (in questo caso MovieClip) possono aggiungere proprietà dinamiche e accedervi in fase di runtime. Per ulteriori informazioni sull'istruzione `dynamic`, vedere `dynamic` statement nella *Guida di riferimento di ActionScript 2.0*.

Identificazione di istanze dinamiche e contenuto caricato

È possibile creare un oggetto tramite ActionScript e identificarlo utilizzando un percorso target in seguito. Il seguente codice ActionScript può ad esempio essere utilizzato per creare un clip filmato. La rotazione del clip filmato può quindi essere modificata sempre utilizzando ActionScript, come illustrato nell'esempio seguente:

Per identificare un'istanza di clip filmato creata in modo dinamico:

1. Creare un nuovo documento Flash e salvarlo come **targetClip.fla**.
2. Inserire un nuovo livello e rinominarlo **actions**.
3. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
this.createEmptyMovieClip("rotateClip", this.getNextHighestDepth());  
trace(rotateClip);  
rotateClip._rotation = 50;
```

4. Selezionare Controllo > Prova filmato per provare il documento.

L'istruzione `trace` indica che è stato creato un clip filmato, ma sullo stage non viene visualizzato alcun elemento. Sebbene sia stato aggiunto codice che crea un'istanza di clip filmato, se non si aggiungono elementi al clip filmato, sullo stage non viene visualizzato niente. Provare ad esempio a caricare un'immagine nel clip filmato.

5. Tornare all'ambiente di creazione e aprire il pannello Azioni.

6. Immettere il codice ActionScript seguente dopo il codice aggiunto al punto 3:

```
rotateClip.loadMovie("http://www.helpexamples.com/flash/images/  
image1.jpg");
```

Questo codice carica un'immagine nel clip filmato `rotateClip` creato tramite codice. Si identifica l'istanza `rotateClip` con ActionScript.

7. Selezionare Controllo > Prova filmato per provare il documento.

Viene visualizzata un'immagine sullo stage che ruota di 50° in senso orario.

È inoltre possibile identificare parti di file SWF caricate in un file SWF di base.

Per identificare un file SWF caricato:

- Utilizzare `_level X` , dove X è il numero di livello specificato nella funzione `loadMovie()` che ha caricato il file SWF.

Un file SWF caricato nel livello 99, ad esempio, ha il percorso target `_level99`.

Nell'esempio seguente, un file SWF viene caricato nel livello 99 e la relativa visibilità viene impostata su `false`:

```
// Carica il file SWF al livello 99.  
loadMovieNum("contents.swf", 99);  
// Imposta la visibilità del livello 99 su false.  
loaderClip.onEnterFrame = function(){  
    _level99._visible = false;  
};
```

SUGGERIMENTO È generalmente consigliabile evitare l'utilizzo di livelli se è possibile caricare il contenuto in clip filmato a profondità diverse. Utilizzando il metodo `MovieClip.getNextHighestDepth()` è possibile creare dinamicamente nuove istanze di clip filmato sullo stage, senza essere costretti a controllare l'esistenza di un'istanza a una determinata profondità.

Impostazione di variabili utilizzando un percorso

È possibile impostare variabili per istanze nidificate all'interno di altre istanze. Se ad esempio si desidera impostare una variabile per un form presente all'interno di un altro form, è possibile utilizzare il codice seguente. L'istanza `submitBtn` si trova all'interno di `formClip` sulla linea temporale principale:

```
this.formClip.submitBtn.mouseOver = true;
```

Per esprimere un metodo o una proprietà di un determinato oggetto, ad esempio un clip filmato o un campo di testo, è possibile utilizzare questo modello. La proprietà di un oggetto, ad esempio, sarebbe

```
myClip._alpha = 50;
```

Informazioni su area di validità e identificazione

Quando si nidificano istanze, il clip filmato che nidifica un secondo clip filmato è detto *elemento principale* dell'istanza nidificata. L'istanza nidificata è detta istanza secondaria. Lo stage principale e la linea temporale principale sono in pratica un clip filmato e pertanto possono essere identificati come tali. Per ulteriori informazioni sull'area di validità, vedere “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362.

È possibile identificare istanze principali e linee temporali principali tramite ActionScript. Per identificare la linea temporale corrente, utilizzare la parola chiave `this`. Per identificare ad esempio un clip filmato denominato `myClip` sulla linea temporale principale, utilizzare:

```
this.myClip.
```

Se si desidera, è possibile evitare la parola chiave `this` e utilizzare esclusivamente `myClip`.

La parola chiave `this` aiuta tuttavia a migliorare la leggibilità e la coerenza del codice. Per ulteriori informazioni sulle linee guida consigliate per la scrittura di codice, consultare il [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0”](#) a pagina 791.

Se si traccia il clip filmato in relazione a uno dei frammenti di codice riportati in precedenza, nel pannello Output viene visualizzato `_level0.myClip`. Se è presente codice ActionScript all'interno del clip filmato `myClip`, ma si desidera identificare la linea temporale principale, identificare l'elemento principale del clip filmato, ovvero lo stage principale. Fare doppio clic su un clip filmato e immettere il seguente codice ActionScript nella linea temporale del clip filmato:

```
trace("me: " + this);
trace("my parent: " + this._parent);
```

Provare il file SWF. Viene visualizzato il messaggio seguente nel pannello Output:

```
me: _level0.myClip
my parent: _level0
```

Il messaggio indica che è stata identificata la linea temporale principale. È possibile utilizzare `parent` per creare il percorso relativo di un oggetto. Se, ad esempio, il clip filmato `dogClip` è nidificato all'interno del clip filmato animato `animalClip`, l'istruzione seguente immessa nell'istanza `dogClip` indica ad `animalClip` di interrompere l'animazione:

```
this._parent.stop();
```

Chi ha dimestichezza con Flash e ActionScript probabilmente conosce già l'uso dell'area di validità `_root`. Quest'area di validità in genere si riferisce alla linea temporale principale del documento Flash corrente. Evitare di utilizzare l'area di validità `_root` se non è strettamente necessario. Al suo posto è possibile utilizzare percorsi target relativi.

Se nel codice si utilizza `_root`, si potrebbero verificare errori in caso di caricamento del file SWF in un altro documento Flash. Quando il file SWF viene caricato in un altro file SWF, `_root` nel file caricato potrebbe puntare all'area di validità principale del file SWF in cui il file è caricato, anziché alla propria area di validità principale come previsto. Per questo motivo si possono verificare risultati imprevisti oppure il file potrebbe non funzionare.

Uso del pulsante Percorso target

A volte non è semplice risalire a un determinato percorso target oppure individuare il percorso target necessario per una porzione di codice. Se si desidera identificare un'istanza presente sullo stage, è possibile utilizzare il pulsante Percorso target per determinare il percorso dell'istanza.

Per utilizzare il pulsante Percorso target:

1. Aprire il pannello Azioni (Finestra > Azioni) e fare clic sul pulsante Inserisce un percorso target. I clip filmato nel documento corrente vengono visualizzati in una finestra di dialogo.
2. Selezionare un'istanza dall'elenco della finestra di dialogo.
3. Fare clic su OK.
4. Il percorso target per l'istanza selezionata viene visualizzato nel pannello Script.

Informazioni sulla sintassi della barra

In Flash 3 e 4 la barra inclinata indicava il percorso target di un clip filmato o di una variabile. Questa sintassi è supportata da ActionScript 1.0 in Flash Player 7 e nelle versioni precedenti ma non in ActionScript 2.0 e Flash Player 7 o Flash Player 8.

L'uso della sintassi della barra è consigliato solo se non è possibile ricorrere ad altre possibilità, ad esempio nel caso in cui si crei contenuto destinato in modo specifico a Flash Player 4 o Flash Lite 1.1 e versioni precedenti, dove l'uso della sintassi della barra è obbligatorio. Per ulteriori informazioni su Flash Lite, vedere la [pagina Web di Flash Lite](#).

Informazioni sui segni di punteggiatura utilizzati nel linguaggio

In Flash sono disponibili molti *segni di punteggiatura*. I più comuni sono il punto e virgola (;), i due punti (:), le parentesi [()] e le parentesi graffe ({ }). Ognuno di essi ha un significato particolare nel linguaggio di Flash e consente di definire i tipi di dati, di terminare le istruzioni o la struttura di ActionScript. Le sezioni seguenti illustrano l'uso dei segni di punteggiatura nel codice.

Per ulteriori informazioni sui segni di punteggiatura utilizzati nel linguaggio, consultare i seguenti argomenti:

- “Punto e virgola e due punti” a pagina 88
- “Parentesi graffe” a pagina 89
- “Parentesi” a pagina 93
- “Informazioni sui valori letterali” a pagina 94
- “Informazioni sui commenti” a pagina 96

Per ulteriori informazioni sull'operatore punto (.) e gli operatori di accesso agli array ([]), vedere “Uso dell'operatore punto e dell'operatore di accesso agli array” a pagina 151. Per ulteriori informazioni sugli spazi vuoti e sulla formattazione del codice, vedere “Formattazione della sintassi ActionScript” a pagina 825.

Punto e virgola e due punti

Le istruzioni ActionScript terminano con il carattere del punto e virgola (;), come illustrato nelle due righe di codice seguenti:

```
var myNum:Number = 50;  
myClip._alpha = myNum;
```

Se si omette il carattere del punto e virgola, il compilatore di ActionScript presuppone che ogni riga di codice rappresenti un'istruzione singola, tuttavia è buona norma utilizzare i punti e virgola per rendere il codice più leggibile. Quando si fa clic sul pulsante Formattazione automatica nel pannello Azioni o nella finestra Script, alla fine di ogni istruzione viene aggiunto per impostazione predefinita un punto e virgola.

NOTA

L'uso di un punto e virgola per terminare un'istruzione consente di posizionare più di un'istruzione su una sola riga. Tuttavia il codice risulta di più difficile lettura.

I punti e virgola vengono utilizzati anche nei cicli `for` per separare i parametri, come nell'esempio seguente. L'esempio esegue iterazioni da 0 a 9 e quindi visualizza ogni numero nel pannello Output:

```
var i:Number;  
for (i = 0; i < 10; i++) {  
    trace(i); // 0,1,...,9  
}
```

I due punti (:) nel codice consentono di assegnare tipi di dati alle variabili. Per assegnare un tipo di dati specifico a un elemento, specificarne il tipo tramite la parola chiave `var` e la sintassi che segue i due punti, come nell'esempio seguente:

```
// Tipizzazione forte dei dati di un oggetto  
var myNum:Number = 7;  
var myDate:Date = new Date();  
// Tipizzazione forte dei dati dei parametri  
function welcome(firstName:String, myAge:Number) {  
}  
// Tipizzazione forte dei dati del parametro e del valore restituito  
function square(num:Number):Number {  
    var squared:Number = num * num;  
    return squared;  
}
```

È possibile dichiarare il tipo di dati degli oggetti in base alle classi incorporate (`Button`, `Date`, `MovieClip` e così via) e alle classi e alle interfacce create dal programmatore. Nel frammento di codice seguente si crea un nuovo oggetto del tipo personalizzato `Student`:

```
var firstStudent:Student = new Student();
```

È possibile inoltre specificare che gli oggetti sono del tipo `Function` o `Void`. Per ulteriori informazioni sull'assegnazione di tipi di dati, consultare il [Capitolo 10, “Dati e tipi di dati” a pagina 333](#).

Parentesi graffe

Gli eventi, le definizioni di classi e le funzioni ActionScript possono essere raggruppati in blocchi tramite le parentesi graffe ({}). La parentesi graffa aperta viene inserita sulla stessa riga della dichiarazione.

NOTA

La parentesi graffa aperta può essere inserita anche sulla riga seguente alla dichiarazione. Le convenzioni per la scrittura di codice suggeriscono di inserirla nella stessa riga per garantire la coerenza. Per informazioni sulle parentesi graffe e le convenzioni per la scrittura di codice, consultare il [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0” a pagina 791](#).

Inserire le parentesi graffe prima e dopo ogni istruzione quando questa fa parte di una struttura di controllo (ad esempio `if..else` o `for`), anche se la struttura contiene una sola istruzione. Questa abitudine aiuta a evitare errori nel codice se si dimentica di inserire le parentesi graffe. Di seguito è riportato un esempio di codice scritto con formato scorretto:

```
var numUsers:Number;  
if (numUsers == 0)  
    trace("no users found.");
```

Sebbene il codice non generi errori, il formato non è corretto perché non sono presenti le parentesi graffe prima e dopo le istruzioni.

SUGGERIMENTO

Le parentesi vengono aggiunte a questa istruzione se si fa clic sul pulsante Controlla sintassi.

In questo caso, se viene aggiunta una seconda istruzione dopo l'istruzione `trace`, la seconda istruzione viene eseguita indipendentemente dal fatto che la variabile `numUsers` sia uguale a 0 e i risultati potrebbero essere imprevisti. Per questo motivo aggiungere le parentesi graffe nell'esempio che segue:

```
var numUsers:Number;  
if (numUsers == 0) {  
    trace("no users found");  
}
```

Nell'esempio seguente vengono creati un oggetto listener di eventi e un'istanza `MovieClipLoader`.

```
var imgUrl:String = "http://www.helpexamples.com/flash/images/image1.jpg";  
this.createEmptyMovieClip("img_mc", 100);  
var mcListener:Object = new Object();  
mcListener.onLoadStart = function() {  
    trace("starting");  
};  
mcListener.onLoadInit = function(target_mc:MovieClip):Void {  
    trace("success");  
};  
mcListener.onLoadError = function(target_mc:MovieClip):Void {  
    trace("failure");  
};  
var myClip1:MovieClipLoader = new MovieClipLoader();  
myClip1.addListener(mcListener);  
myClip1.loadClip(imgUrl, img_mc);
```

L'esempio seguente include un file di classe semplice che può essere utilizzato per creare un oggetto Student. Per ulteriori informazioni sui file di classe, consultare il [Capitolo 6, “Classi”](#) a pagina 197.

Per utilizzare parentesi graffe in un file di ActionScript:

1. Selezionare File > Nuovo, quindi selezionare File ActionScript.
2. Selezionare File > Salva con nome e salvare il nuovo documento come **Student.as**.
3. Aggiungere al file AS il codice ActionScript seguente:

```
// Student.as
class Student {
    private var _id:String;
    private var _firstName:String;
    private var _middleName:String;
    private var _lastName:String;

    public function Student(id:String, firstName:String,
        middleName:String, lastName:String) {
        this._id = id;
        this._firstName = firstName;
        this._middleName = middleName;
        this._lastName = lastName;
    }
    public function get firstName():String {
        return this._firstName;
    }
    public function set firstName(value:String):Void {
        this._firstName = value;
    }
    // ...
}
```

4. Salvare il file di classe.
5. Selezionare File > Nuovo e fare clic su Documento Flash per creare un nuovo file FLA.
6. Salvare il nuovo file FLA come **student_test.fla**.
7. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale principale:

```
// student_test.fla
import Student;
var firstStudent:Student = new Student("cst94121", "John", "H.", "Doe");
trace(firstStudent.firstName); // John
firstStudent.firstName = "Craig";
trace(firstStudent.firstName); // Craig
```

8. Per salvare le modifiche a **student_test.fla**, selezionare File > Salva.
9. Selezionare Controllo > Prova filmato per provare il file FLA e il file AS.

L'esempio seguente dimostra l'uso delle parentesi graffe con le funzioni.

Per utilizzare le parentesi graffe con le funzioni:

1. Selezionare File > Nuovo, quindi Documento Flash per creare un nuovo file FLA.
2. Selezionare File > Salva con nome e assegnare al nuovo file il nome **checkform.fla**.
3. Trascinare un'istanza del componente Label dal pannello Componenti nello stage.
4. Aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà) e con l'istanza del componente Label selezionata digitare il nome di istanza **status_lbl** nella casella di testo Nome istanza.
5. Digitare **200** nella casella di testo relativa alla larghezza per ridimensionare il componente a 200 pixel.
6. Trascinare un'istanza del componente TextInput nello stage e assegnarvi il nome di istanza **firstName_ti**.
7. Trascinare un'istanza del componente Button nello stage e assegnarvi il nome di istanza **submit_button**.
8. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```
function checkForm():Boolean {  
    status_lbl.text = "";  
    if (firstName_ti.text.length == 0) {  
        status_lbl.text = "Please enter a first name.";  
        return false;  
    }  
    return true;  
}  
function clickListener(evt_obj:Object):Void {  
    var success:Boolean = checkForm();  
};  
submit_button.addEventListener("click", clickListener);
```

9. Selezionare File > Salva per salvare il documento Flash.
10. Selezionare Controllo > Prova filmato per provare il codice nell'ambiente di creazione.

Nel file SWF viene visualizzato un messaggio di errore se si fa clic sull'istanza del pulsante Button sullo stage quando non è presente testo nel componente **firstName_ti** TextInput. Il messaggio viene visualizzato nel componente Label e informa gli utenti che devono immettere un nome.

Il prossimo esempio sull'uso delle parentesi graffe illustra come creare e definire proprietà all'interno di un oggetto. Le proprietà vengono definite nell'oggetto specificando i nomi delle variabili all'interno delle parentesi graffe ({}):

```
var myObject:Object = {id:"cst94121", firstName:"John", middleName:"H.",  
    lastName:"Doe"};  
var i:String;  
for (i in myObject) {  
    trace(i + ": " + myObject[i]);  
}  
/*  
 id: cst94121  
 firstName: John  
 middleName: H.  
 lastName: Doe  
*/
```

È anche possibile utilizzare parentesi graffe vuote come una sintassi abbreviata per la funzione new Object(). Il codice seguente, ad esempio, crea un'istanza vuota di Object:

```
var myObject:Object = {};
```

SUGGERIMENTO Controllare sempre che a ogni parentesi graffa aperta corrisponda una parentesi graffa chiusa.

Parentesi

Per definire una funzione in ActionScript, inserire i parametri all'interno di parentesi [()], come illustrato nelle righe di codice seguenti:

```
function myFunction(myName:String, myAge:Number, happy:Boolean):Void {  
    // Inserire qui il codice  
}
```

Per chiamare una funzione, racchiudere anche i parametri passati alla funzione tra parentesi, come nell'esempio seguente:

```
myFunction("Carl", 78, true);
```

È inoltre possibile utilizzare le parentesi tonde per sostituire l'ordine di priorità di ActionScript o per migliorare la leggibilità delle istruzioni ActionScript. Per modificare l'ordine di elaborazione dei valori, inserire determinati valori tra parentesi, come nell'esempio seguente:

```
var computedValue:Number = (circleClip._x + 20) * 0.8;
```

A causa dell'ordine di priorità, se non si utilizzassero parentesi o si utilizzassero due istruzioni separate, la moltiplicazione verrebbe elaborata per prima e quindi la prima operazione sarebbe $20 * 0.8$. Il risultato, 16, verrebbe aggiunto al valore corrente di circleClip._x e infine assegnato alla variabile computedValue.

Se non si utilizzano le parentesi, aggiungere un'istruzione per valutare l'espressione, come nell'esempio seguente:

```
var tempValue:Number = circleClip._x + 20;  
var computedValue:Number = tempValue * 0.8;
```

Come con le parentesi quadre e graffe, verificare che a ogni parentesi aperta corrisponda una parentesi chiusa.

Informazioni sui valori letterali

Per *valore letterale* si intende un valore che compare direttamente nel codice. I valori letterali sono valori costanti, ovvero invariabili, all'interno dei documenti Flash. Comprendono true, false, 0, 1, 52 o la stringa "foo".

I seguenti esempi sono valori letterali:

```
17  
"hello"  
-3  
9.4  
null  
undefined  
true  
false
```

I valori letterali possono essere raggruppati a formare valori letterali composti. I valori letterali array sono racchiusi tra parentesi quadre ([]) e utilizzano la virgola (,) per separare gli elementi dell'array. Un valore letterale array può essere utilizzato per inizializzare un array. Gli esempi seguenti mostrano due array inizializzati tramite valori letterali array. È possibile utilizzare l'istruzione new e passare il valore letterale composto come parametro alla funzione di costruzione della classe Array, ma è anche possibile assegnare valori letterali direttamente durante la creazione di istanze di una classe ActionScript incorporata.

```
// Viene utilizzata l'istruzione new
var myStrings:Array = new Array("alpha", "beta", "gamma");
var myNums:Array = new Array(1, 2, 3, 5, 8);

// Il valore letterale viene assegnato direttamente
var myStrings:Array = ["alpha", "beta", "gamma"];
var myNums:Array = [1, 2, 3, 5, 8];
```

I valori letterali possono essere anche utilizzati per inizializzare un oggetto generico, ovvero un'istanza della classe Object. I valori letterali oggetto sono racchiusi tra parentesi graffe ({}) e utilizzano la virgola (,) per separare le proprietà dell'oggetto. Ogni proprietà viene dichiarata con i due punti (:) che separano il nome della proprietà dal relativo valore.

È possibile creare un oggetto generico utilizzando l'istruzione new e passando il valore letterale oggetto come parametro alla funzione di costruzione della classe Object oppure assegnare il valore letterale oggetto direttamente all'istanza che si dichiara. L'esempio seguente crea un nuovo oggetto generico e lo inizializza con tre proprietà propA, propB e propC, con valori impostati rispettivamente su 1, 2 e 3.

```
// Viene utilizzata l'istruzione new
var myObject:Object = new Object({propA:1, propB:2, propC:3});

// Il valore letterale viene assegnato direttamente
var myObject:Object = {propA:1, propB:2, propC:3};
```

È importante non confondere un carattere letterale di stringa con un oggetto String. Nell'esempio seguente, la prima riga di codice crea il carattere letterale di stringa first_string, mentre la seconda riga crea l'oggetto String secondStr:

```
var firstStr:String = "foo"
var secondStr:String = new String("foo")
```

Utilizzare i caratteri letterali di stringa a meno che non sia espressamente necessario utilizzare un oggetto String per garantire prestazioni migliori. Per ulteriori informazioni sulle stringhe, consultare ["Informazioni sulle stringhe e sulla classe String"](#) a pagina 492.

Informazioni sui commenti

I commenti consentono di inserire annotazioni nel codice con descrizioni nella lingua desiderata, ad esempio in italiano, che non vengono valutate dal compilatore. Possono essere utilizzati nel codice per descrivere l'azione eseguita in quel punto oppure i dati restituiti. Possono inoltre aiutare a ricordare il motivo per cui è stata scelta una determinata soluzione nel codice e sono utili per gli utenti che leggono il codice. Devono descrivere in modo chiaro lo scopo del codice e non limitarsi a tradurre il codice. Se un'operazione non risulta immediata leggendo il codice, aggiungere commenti per illustrarla.

Per aggiungere note agli script, è consigliato l'utilizzo di commenti. I commenti documentano le decisioni prese nel codice e devono spiegare come e perché è stata eseguita una determinata scelta. Facilitano inoltre la comprensione del codice ActionScript e potrebbero descrivere, ad esempio, una soluzione alternativa. In questo modo è più facile anche per altri sviluppatori trovare le sezioni di codice da aggiornare o correggere oppure, se il problema viene risolto o la funzionalità migliorata in una versione futura di Flash o Flash Player, è possibile migliorare il codice ActionScript rimuovendo la soluzione alternativa.

Non aggiungere commenti disordinati, ad esempio righe di segni uguale (=) o asterischi (*) per creare un blocco o una separazione tra il commento e il codice. Utilizzare invece spazi vuoti per separare i commenti dal codice ActionScript. Se il codice ActionScript viene formattato tramite il pulsante Formattazione automatica del pannello Azioni, lo spazio vuoto viene rimosso. Ricordarsi di aggiungere nuovamente lo spazio nel codice oppure utilizzare righe di commenti singole (//) per gestire la spaziatura. È più semplice rimuovere queste righe dopo la formattazione del codice che tentare di determinare dove si trovasse lo spazio vuoto.

Prima di distribuire il progetto, rimuovere tutti i commenti superflui dal codice, ad esempio "definire le variabili x e y" o altri commenti non utili per altri sviluppatori. Se nel codice ActionScript sono presenti troppi commenti, provare a riscrivere parte del codice. Se si sente la necessità di inserire molti commenti sul funzionamento del codice, in genere significa che il codice ActionScript non è scritto in modo chiaro e il suo funzionamento non è intuitivo.

Se viene attivata la colorazione sintassi, i commenti sono visualizzati in grigio per impostazione predefinita. La lunghezza del commento non influisce sulle dimensioni del file esportato; inoltre non occorre che i commenti seguano le regole della sintassi e delle parole chiave di ActionScript.

NOTA

L'utilizzo di commenti in ActionScript è particolarmente importante se si utilizza il codice a scopo didattico. Aggiungere commenti al codice se si creano applicazioni di esempio per insegnare a utilizzare Flash oppure se si scrivono articoli o esercitazioni su ActionScript.

Commenti a riga singola

I commenti a riga singola consentono di aggiungere un commento su una sola riga di codice. È possibile impostare una riga di codice come commento o aggiungere una breve descrizione dell'azione eseguita da quella porzione di codice. Per indicare che una riga, o parte di essa, è un commento, vengono anteposte due barre (//), come nel codice seguente:

```
// Imposta una variabile locale per l'età  
var myAge:Number = 26;
```

I commenti a riga singola vengono in genere utilizzati per descrivere un frammento di codice di piccole dimensioni e può essere utilizzato per tutti i commenti di lunghezza limitata a su una sola riga. L'esempio seguente comprende un commento a riga singola:

```
while (condition) {  
    // Gestire la condizione con istruzioni  
}
```

Commenti su più righe

I commenti su più righe, detti anche commenti a blocchi, vengono utilizzati per commenti di lunghezza superiore a una riga. Questo tipo di commenti viene in genere utilizzato dagli sviluppatori per descrivere file, strutture di dati, metodi e consente di fornire descrizioni di file. Vengono generalmente inseriti all'inizio di un file e prima di un metodo o al suo interno.

Per creare un blocco di commenti, posizionare /* all'inizio delle righe di commento e */ alla fine del blocco. In questo modo è possibile creare commenti piuttosto lunghi senza dover iniziare ogni riga con //. L'uso di // per diverse righe di seguito può creare problemi al momento della modifica dei commenti.

Il formato di un commento su più righe è analogo al seguente:

```
/*  
 Il codice ActionScript seguente inizializza le variabili utilizzate nei  
 menu principali e secondari che consentono di tenere traccia delle  
 opzioni su cui l'utente fa clic.  
*/
```

SUGGERIMENTO Se i caratteri che identificano il commento /* e */) vengono inseriti su righe separate all'inizio e alla fine del commento, è facile impostarli a loro volta come commenti, facendoli precedere da una doppia barra (//), ad esempio /** e **/. Questa procedura consente di impostare in modo semplice e rapido il codice come commento ed eliminare i caratteri di commento quando necessario.

È possibile impostare ampie porzioni di script come commenti per provare solo determinate parti di uno script. Ad esempio, quando si esegue lo script riportato di seguito, non viene eseguita alcuna parte del codice presente nel blocco di commento.

```
// Il codice seguente viene eseguito.  
var x:Number = 15;  
var y:Number = 20;  
  
// Il codice seguente è commentato e non verrà eseguito.  
/*  
// Crea un nuovo oggetto Date.  
var myDate:Date = new Date();  
var currentMonth:Number = myDate.getMonth();  
// Converte il numero del mese nel relativo nome.  
var monthName:String = calcMonth(currentMonth);  
var year:Number = myDate.getFullYear();  
var currentDate:Number = myDate.getDate();  
*/  
  
// Il codice seguente viene eseguito.  
var namePrefix:String = "My name is";  
var age:Number = 20;
```

SUGGERIMENTO

Si consiglia di far precedere ogni commento a blocchi da una riga vuota.

Commenti finali

Con questo tipo di commenti, il commento viene aggiunto e visualizzato sulla stessa riga del codice ActionScript. In genere questi commenti vengono utilizzati per indicare il contenuto di una variabile o descrivere o commentare il valore restituito da una riga di codice ActionScript. Il formato dei commenti finali è il seguente:

```
var myAge:Number = 26; // Variabile per l'età (myAge)  
trace(myAge); // 26
```

Inserire il commento a destra dopo alcuni spazi per consentire di distinguerlo dal codice. Se possibile, cercare di allineare i commenti, come nel codice seguente:

```
var myAge:Number = 28; // Età  
var myCountry:String = "Canada"; // Paese di appartenenza  
var myCoffee:String = "Hortons"; // Tipo di caffè preferito
```

Se si utilizza il pulsante Formattazione automatica nel pannello Azioni, i commenti finali vengono spostati alla riga successiva. Aggiungere i commenti dopo avere formattato il codice se non si desidera che vengano spostati quando viene utilizzato il pulsante Formattazione automatica.

Commenti all'interno delle classi

È possibile utilizzare commenti in classi ed interfacce per documentarle e aiutare gli sviluppatori a comprenderne il contenuto, ad esempio iniziando tutti i file di classe con un commento che indichi il nome della classe, il numero di versione, la data e il copyright. È possibile, ad esempio, creare documentazione per la classe con un commento simile al seguente:

```
/**  
 * Classe Pelican  
 * Versione 1.2  
 * 10/10/2005  
 * Copyright Macromedia, Inc.  
 */
```

I commenti a blocchi possono essere utilizzati per descrivere file, strutture di dati, metodi e consentono di fornire descrizioni di file. Questi commenti vengono generalmente inseriti all'inizio di un file e prima di un metodo o al suo interno.

In un file di interfaccia o di classe tipico sono presenti due tipi di commenti: commenti di documentazione e commenti di implementazione. I commenti di documentazione vengono utilizzati per descrivere le specifiche del codice e non descrivono l'implementazione. Possono descrivere interfacce, classi, metodi e funzioni di costruzione. I commenti di implementazione, invece, vengono utilizzati per impostare porzioni di codice come commento o inserire commenti sull'implementazione di determinate sezioni di codice.

Includere un unico commento di documentazione per classe, interfaccia o membro, inserendolo subito prima della dichiarazione. Se si desidera inserire ulteriori informazioni di documentazione e lo spazio nei commenti di documentazione non è sufficiente, fare ricorso ai commenti di implementazione, utilizzando il formato dei commenti a blocchi o a riga singola. I commenti di implementazione vengono inseriti subito dopo la dichiarazione.

Per i due tipi di commenti vengono utilizzati delimitatori leggermente diversi: i commenti di documentazione sono delimitati con `/**` e `*/`, mentre i commenti di implementazione sono delimitati con `/*` e `*/`.

SUGGERIMENTO

Non inserire commenti che non si riferiscano direttamente alla classe che viene letta, ad esempio commenti che descrivono il pacchetto corrispondente.

Nei file di classe è inoltre possibile utilizzare commenti a riga singola, commenti a blocchi e commenti finali. Per ulteriori informazioni su questi tipi di commenti, consultare le seguenti sezioni:

- “[Commenti a riga singola](#)” a pagina 97
- “[Commenti su più righe](#)” a pagina 97
- “[Commenti finali](#)” a pagina 98

Informazioni su costanti e parole chiave

Le costanti e le parole chiave sono la base sui cui poggia la sintassi di ActionScript. Le costanti sono proprietà con valore fisso che non possono essere modificate, ovvero valori che non cambiano in tutta l'applicazione.

Flash comprende diverse costanti predefinite che possono aiutare a sviluppare le applicazioni in modo più semplice. Un esempio di costanti è presente nella classe Key che comprende diverse proprietà, ad esempio `Key.ENTER` o `Key.PGDN`. Se si utilizzano le costanti, non occorre ad esempio tenere a mente che i valori di codice per i tasti Invio e Pg giù sono 13 e 34. L'uso di valori costanti non semplifica solo lo sviluppo e il debug, ma anche la lettura del codice per altri sviluppatori.

Le parole chiave in ActionScript vengono utilizzate per eseguire determinati tipi di azioni. Si tratta di parole riservate, pertanto non sono utilizzabili come identificatori, ad esempio nomi di variabili, di funzioni o di etichette. Esempi di alcune parole chiave riservate sono `if`, `else`, `this`, `function` e `return`.

Per ulteriori informazioni sulle costanti e sulle parole chiave, consultare i seguenti argomenti:

- “[Uso delle costanti](#)” a pagina 101
- “[Informazioni sulle parole chiave](#)” a pagina 103
- “[Informazioni sulle parole riservate](#)” a pagina 104

Per ulteriori informazioni su oggetti e proprietà, consultare “[Tipo di dati oggetto](#)” [a pagina 341](#). Per un elenco di costanti del linguaggio, ad esempio `false` e `NAN`, vedere la categoria Elementi del linguaggio ActionScript > Costanti nella *Guida di riferimento di ActionScript 2.0*.

Uso delle costanti

Le costanti sono proprietà con valore fisso che non possono essere modificate, ovvero valori che non cambiano in tutta l'applicazione. Il linguaggio ActionScript comprende molte costanti predefinite. Le costanti `BACKSPACE`, `ENTER`, `SPACE` e `TAB`, ad esempio, sono proprietà della classe `Key` e si riferiscono ai tasti della tastiera. La costante `Key.TAB`, ad esempio, indica sempre lo stesso elemento, ossia il tasto `TAB` presente sulla tastiera. Le costanti sono utili per confrontare valori e utilizzano valori che non cambiano nell'applicazione.

Se si desidera verificare se l'utente preme il tasto Invio, è possibile usare la seguente istruzione:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
    if (Key.getCode() == Key.ENTER) {
        trace("Are you ready to play?");
    }
}
Key.addListener(keyListener);
```

Affinché il codice ActionScript riportato in precedenza funzioni, può essere necessario disabilitare i tasti di scelta rapida nell'ambiente di creazione. Selezionare Controllo > Prova filmato dal menu principale, quindi durante l'anteprima del file SWF nel lettore selezionare Controllo > Disattiva tasti di scelta rapida da tastiera nella finestra di anteprima del file SWF.

In Flash non è possibile creare valori costanti personalizzati se non si creano classi personalizzate con variabili membro private. Non è possibile creare una variabile di sola lettura.

Mentre i nomi delle variabili devono essere composti da lettere minuscole o da parole con iniziale maiuscola, i nomi delle costanti (variabili che non cambiano) devono essere composti da lettere maiuscole. Separare le parole con il carattere di sottolineatura, come illustrato nel seguente codice ActionScript:

```
var BASE_URL:String = "http://www.macromedia.com"; // Costante
var MAX_WIDTH:Number = 10; // Costante
```

Scrivere le costanti statiche a lettere maiuscole e separare le parole con un carattere di sottolineatura. Non scrivere direttamente costanti numeriche a meno che non si tratti di 1, 0 o -1, utilizzabili in un ciclo `for` come valori di contatore.

Le costanti possono essere utilizzate quando occorre fare riferimento a una proprietà il cui valore non cambia mai. In questo modo è più semplice trovare errori di ortografia nel codice, che sarebbero invece difficili da rilevare utilizzando caratteri letterali, ed è possibile modificare il valore in un solo punto. Per ulteriori informazioni sui valori letterali, vedere “[Informazioni sui valori letterali](#)” a pagina 94.

La definizione di classe dell'esempio seguente, ad esempio, crea tre costanti che seguono la convenzione di denominazione di ActionScript 2.0.

Per utilizzare costanti in un'applicazione:

1. Selezionare File > Nuovo e quindi selezionare File ActionScript per creare un file AS.
2. Assegnare al nuovo file il nome **ConstExample.as**.
3. Immettere il codice seguente nella finestra Script:

```
class ConstExample {  
    public static var EXAMPLE_STATIC:String = "Global access";  
    public var EXAMPLE_PUBLIC:String = "Public access";  
    private var EXAMPLE_PRIVATE:String = "Class access";  
}
```

La proprietà EXAMPLE_STATIC è statica ovvero viene applicata a tutta la classe e non a un'istanza particolare della classe. Per accedere a una proprietà statica di una classe è necessario utilizzare il nome della classe e non il nome di un'istanza. Non è possibile accedere a una proprietà statica tramite un'istanza di classe.

4. Creare un nuovo documento Flash e salvarlo come **const.fla**.
5. Aprire il pannello Azioni e digitare il codice seguente nel fotogramma 1 della linea temporale:

```
trace(ConstExample.EXAMPLE_STATIC); // Output: Global access
```

Se si dichiara la proprietà EXAMPLE_STATIC come statica, è necessario accedere al relativo valore tramite questo codice.
6. Selezionare Controllo > Prova filmato per provare il documento.
Nel pannello Output viene visualizzato Global access.
7. Sempre nel pannello Azioni, aggiungere il codice seguente di seguito a quello aggiunto al punto 5.

```
trace(ConstExample.EXAMPLE_PUBLIC); // Errore  
trace(ConstExample.EXAMPLE_PRIVATE); // Errore
```
8. Selezionare Controllo > Prova filmato per provare il documento.
Le proprietà EXAMPLE_PUBLIC e EXAMPLE_PRIVATE non sono statiche. Quando si tenta di accedere ai loro valori tramite la classe, viene visualizzato il seguente messaggio di errore:
The property being referenced does not have the static attribute.

Per accedere a una proprietà non statica, è necessario accedere al valore tramite un'istanza della classe. Dato che la proprietà EXAMPLE_PUBLIC è pubblica, è accessibile con codice esterno alla definizione di classe.

9. Nel pannello Azioni eliminare le istruzioni trace aggiunte ai punti 5 e 7.

10. Immettere il codice seguente nel pannello Azioni:

```
var myExample:ConstExample = new ConstExample();
trace(myExample.EXAMPLE_PUBLIC); // Output: Public access
```

Questo codice crea un'istanza dell'istanza myExample e accede alla proprietà EXAMPLE_PUBLIC.

11. Selezionare Controllo > Prova filmato per provare il documento.

Nel pannello Output viene visualizzato Public access.

12. Nel pannello Azioni eliminare l'istruzione trace aggiunta al punto 10.

13. Immettere il codice seguente nel pannello Azioni.

```
trace(myExample.EXAMPLE_PRIVATE); // Errore
```

La proprietà EXAMPLE_PRIVATE è privata, pertanto accessibile solo all'interno della definizione di classe.

14. Selezionare Controllo > Prova filmato per provare il documento.

Nel pannello Output viene visualizzato Il membro è privato. Impossibile accedervi.

Per ulteriori informazioni sulle classi incorporate e sulla creazione di classi personalizzate, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Informazioni sulle parole chiave

Le parole chiave sono parole utilizzate in ActionScript per scopi particolari. Ad esempio, si utilizza la variabile var per dichiarare una variabile. La parola chiave var è illustrata nella riga di codice seguente:

```
var myAge:Number = 26;
```

Una parola chiave è una parola riservata con un significato particolare, ad esempio è possibile utilizzare la parola chiave class per definire una nuova classe ActionScript e la parola chiave var per dichiarare variabili locali. Altri esempi di parole chiave riservate sono: if, else, this, function e return.

Le parole chiave non possono essere utilizzate come identificatori, ad esempio variabili, funzioni o nomi di etichetta e non devono essere utilizzate in altri punti dei file FLA per scopi diversi, ad esempio come nomi di istanza. La parola chiave `var` è già stata utilizzata molte volte, in particolare nel [Capitolo 10, “Dati e tipi di dati” a pagina 333](#). In ActionScript alcune parole del linguaggio sono riservate per usi specifici, pertanto non possono essere utilizzate come identificatori, ovvero nomi di variabili, funzioni o etichette. È possibile trovare un elenco di queste parole chiave in [“Informazioni sulle parole riservate” a pagina 104](#).

Informazioni sulle parole riservate

Per *parole riservate* si intendono parole che non possono essere utilizzate come identificatori nel codice perché sono riservate per l'uso in ActionScript. Includono le *parole chiave*, vale a dire istruzioni ActionScript, e parole riservate per uso futuro. Non possono quindi essere utilizzate come nomi per variabili, istanze, classi personalizzate e così via. In caso contrario, si verificherebbero problemi tecnici.

Nella tabella seguente sono elencate le parole riservate di Flash che causano errori di script:

add	and	break	case
catch	class	continue	default
delete	do	dynamic	else
eq	extends	finally	for
function	ge	get	gt
if	ifFrameLoaded	implements	import
in	instanceof	interface	intrinsic
le	lt	ne	new
not	on	onClipEvent	or
private	public	return	set
static	switch	tellTarget	this
throw	try	typeof	var
void	while	with	

Nella tabella seguente sono elencate le parole chiave riservate per uso futuro in ActionScript o dalla bozza di proposta ECMAScript (ECMA-262) edizione 4 di specifica del linguaggio.
Non utilizzare queste parole chiave nel codice:

abstract	enum	export	short
byte	long	synchronized	char
debugger	protected	double	volatile
float	throws	transient	goto

Tutti i nomi di classi incorporate, i nomi di classi di componenti e i nomi di interfaccia sono parole riservate e non devono essere utilizzate come identificatori nel codice:

Accessibility	Accordion	Alert	Array
Binding	Boolean	Button	Camera
CellRenderer	CheckBox	Collection	Color
ComboBox	ComponentMixins	ContextMenu	ContextMenuItem
CustomActions	CustomFormatter	CustomValidator	DataGrid
DataHolder	DataProvider	DataSet	DataType
Date	DateChooser	DateField	Delta
DeltaItem	DeltaPacket	DepthManager	EndPoint
Error	FocusManager	Form	Function
Iterator	Key	Label	List
Loader	LoadVars	LocalConnection	Log
Math	Media	Menu	MenuBar
Microphone	Mouse	MovieClip	MovieClipLoader
NetConnection	NetStream	Number	NumericStepper
Object	PendingCall	PopUpManager	PrintJob
ProgressBar	RadioButton	RDBMSResolver	Screen
ScrollPane	Selection	SharedObject	Slide
SOAPCall	Sound	Stage	String
StyleManager	System	TextArea	TextField
TextFormat	TextInput	TextSnapshot	TransferObject
Tree	TreeDataProvider	TypedValue	UIComponent
UIEventDispatcher	UIObject	Video	WebService

WebServiceConnector	Window	XML	XMLConnector
XUpdateResolver			

Diverse parole, pur non essendo parole riservate, non dovrebbero essere utilizzate come identificatori, ovvero nomi di istanze o variabili, nel codice ActionScript. Si tratta di parole utilizzate dalle classi incorporate del linguaggio ActionScript. Non utilizzare pertanto i nomi delle proprietà, dei metodi, delle classi, delle interfacce, delle classi dei componenti e i valori per assegnare nomi a variabili, classi o istanze nel codice.

Per informazioni su questi nomi, consultare la *Guida di riferimento di ActionScript 2.0* e nel pannello della Guida, in *Apprendimento di ActionScript 2.0 in Flash*, consultare le sezioni relative all'uso del linguaggio.

Informazioni sulle istruzioni

Un'*istruzione* è come un comando impartito al file FLA affinché venga eseguita un'operazione, ad esempio eseguire una determinata azione. È possibile ad esempio fare ricorso a un'istruzione condizionale per determinare se qualcosa è vero o esiste e quindi eseguire determinate azioni, ad esempio funzioni o espressioni, in base al fatto che la condizione sia soddisfatta o meno.

L'istruzione `if`, ad esempio, è un'istruzione condizionale e restituisce una condizione con lo scopo di determinare l'azione successiva che deve essere eseguita nel codice.

```
// Istruzione if
if (condition) {
    // Istruzioni;
}
```

Un ulteriore esempio è rappresentato dall'istruzione `return` che restituisce un risultato sotto forma di valore della funzione in cui viene eseguita.

Il codice ActionScript può essere scritto o formattato in modo diverso. La sintassi del codice ActionScript può essere formattata in modo diverso da diversi programmati, in particolare per quanto riguarda la spaziatura delle istruzioni o l'inserimento delle parentesi graffe (`{ }`). Sebbene questi diversi formati delle istruzioni non influiscano sul funzionamento del codice, si consiglia di seguire alcune linee guida generali per scrivere codice ActionScript ordinato e corretto.

In primo luogo, per migliorare la leggibilità inserire una sola istruzione per riga. L'esempio seguente illustra un uso corretto e un uso errato di un'istruzione:

```
theNum++;      // Consigliato
theOtherNum++; // Consigliato
aNum++; anOtherNum++; // Sconsigliato
```

Assegnare le variabili in istruzioni separate. Osservare l'esempio di codice seguente:

```
var myNum:Number = (a = b + c) + d;
```

In questo codice un'assegnazione è incorporata nel codice e pertanto difficile da leggere. Se le variabili vengono assegnate in istruzioni separate, la leggibilità ne risulta migliorata, come illustrato nell'esempio seguente:

```
var a:Number = b + c;  
var myNum:Number = a + d;
```

Le sezioni seguenti illustrano come formattare istruzioni specifiche di ActionScript. Per informazioni sulla scrittura e la formattazione di eventi, consultare il [Capitolo 9, “Gestione degli eventi” a pagina 311](#).

Per ulteriori informazioni su ogni istruzione, consultare i seguenti argomenti:

- [“Informazioni sulle istruzioni composte” a pagina 107](#)
- [“Informazioni sulle condizioni” a pagina 108](#)
- [“Come ripetere azioni utilizzando i cicli” a pagina 119](#)

Informazioni sulle istruzioni composte

Un'istruzione composta contiene numerose istruzioni che possono essere racchiuse tra parentesi graffe ({}). Le istruzioni all'interno di un'istruzione composta possono essere rappresentate da qualsiasi istruzione ActionScript. Di seguito è riportata un'istruzione composta tipica:

Le istruzioni all'interno delle parentesi graffe sono rientrate rispetto all'istruzione composta, come nel codice seguente:

```
var a:Number = 10;  
var b:Number = 10;  
if (a == b) {  
    // Questo codice è rientrato  
    trace("a == b");  
    trace(a);  
    trace(b);  
}
```

Questa istruzione composta contiene diverse istruzioni ma si comporta come un'istruzione singola nel codice ActionScript. La parentesi graffa aperta è stata inserita alla fine dell'istruzione composta. La parentesi graffa chiusa è stata inserita all'inizio di una riga ed è allineata con l'inizio dell'istruzione composta.

Per ulteriori informazioni sull'uso delle parentesi graffe, vedere [“Parentesi graffe” a pagina 89](#).

Informazioni sulle condizioni

Le condizioni consentono di determinare se qualcosa è vero o esiste e, se lo si desidera, di ripetere un'azione (tramite cicli) o eseguire le azioni specificate, come funzioni o espressioni, in base al fatto che la condizione sia soddisfatta o meno. È possibile ad esempio determinare se una variabile è definita o associata a un certo valore ed eseguire un blocco di codice in base al risultato. È inoltre possibile modificare la grafica all'interno del documento Flash in base all'ora su cui è impostato l'orologio di sistema dell'utente o alle condizioni meteo in una determinata località.

Per eseguire un'azione in base alla presenza di una condizione o ripetere un'azione (vale a dire creare istruzioni cicliche), è possibile utilizzare le istruzioni `if`, `else`, `else if`, `for`, `while`, `do while`, `for..in` o `switch`.

Per ulteriori informazioni sulle condizioni utilizzabili e su come possono essere scritte, consultare i seguenti argomenti:

- “[Informazioni sulla scrittura di condizioni](#)” a pagina 108
- “[Uso dell'istruzione if](#)” a pagina 109
- “[Uso dell'istruzione if..else](#)” a pagina 110
- “[Uso dell'istruzione if..else if](#)” a pagina 111
- “[Uso di un'istruzione switch](#)” a pagina 113
- “[Uso delle istruzioni try..catch e try..catch..finally](#)” a pagina 115
- “[Informazioni sull'operatore condizionale e sintassi alternativa](#)” a pagina 118

Informazioni sulla scrittura di condizioni

Le istruzioni che verificano se una condizione è true o false iniziano con il termine `if`. Se la condizione restituisce `true`, ActionScript esegue l'istruzione successiva. Se la condizione restituisce `false`, ActionScript passa all'istruzione successiva all'esterno del blocco di codice.

SUGGERIMENTO

Per ottimizzare le prestazioni del codice, è opportuno verificare prima le condizioni più probabili.

Le seguenti istruzioni verificano tre condizioni. Il termine `else if` specifica verifiche alternative da eseguire se le condizioni precedenti risultano false.

```
if ((passwordTxt.text.length == 0) || (emailTxt.text.length == 0)) {  
    gotoAndStop("invalidLogin");  
} else if (passwordTxt.text == userID){  
    gotoAndPlay("startProgram");  
}
```

In questo frammento di codice, se la lunghezza dei campi di testo `passwordTxt` o `emailTxt` è 0 (ad esempio se l'utente non ha immesso valori), il documento Flash esegue il reindirizzamento all'etichetta del fotogramma `invalidLogin`. Se entrambi i campi di testo contengono valori e il contenuto del campo `passwordTxt` corrisponde alla variabile `userID`, il file SWF esegue il reindirizzamento all'etichetta del fotogramma `startProgram`.

Se si desidera verificare una sola condizione tra molte, è possibile utilizzare l'istruzione `switch` anziché più istruzioni `else if`. Per ulteriori informazioni sulle istruzioni `switch`, vedere [“Uso di un'istruzione switch” a pagina 113](#).

Consultare le sezioni seguenti per informazioni su come scrivere diversi tipi di condizioni nelle applicazioni ActionScript.

Uso dell'istruzione if

L'istruzione `if` consente di escludere una serie di istruzioni in base al fatto che una determinata condizione sia soddisfatta.

```
// Istruzione if  
if (condition) {  
    // Istruzioni;  
}
```

Questo tipo di istruzione viene utilizzata frequentemente nei progetti Flash. Se, ad esempio, si crea un sito Flash che richiede che l'utente esegua l'accesso per raggiungere diverse sezioni di un sito Web, un'istruzione `if` permette di controllare che l'utente immetta testo nei campi relativi al nome utente e alla password.

Se i nomi utenti e le password devono essere convalidati a fronte di un database esterno, occorre probabilmente verificare che la combinazione nome utente/password immessa dall'utente corrisponda a un record del database e che l'utente disponga dell'autorizzazione per accedere all'area del sito specificata.

Se si creano script per animazioni in Flash, l'istruzione `if` permette di controllare se un'istanza sullo stage si trova ancora all'interno dei limiti dello stage. Se, ad esempio, un pallone si sposta verso il basso lungo l'asse `y`, potrebbe essere necessario rilevare quando raggiunge il margine inferiore dello stage per poter cambiare direzione e fare in modo che sembri che rimbalzi in avanti.

Per utilizzare un'istruzione if:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
// Crea una stringa per memorizzare AM e PM
var amPm:String = "AM";
// Nessun parametro viene passato a Date e quindi viene restituita la
// data e l'ora corrente
var current_date:Date = new Date();
// Se l'ora corrente è maggiore o uguale a 12, la stringa amPm viene
// impostata su "PM"
if (current_date.getHours() >= 12) {
    amPm = "PM";
}
trace(amPm);
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

In questo codice si crea una stringa che memorizza AM o PM in base all'ora del giorno. Se l'ora corrente è maggiore o uguale a 12, la stringa `amPm` viene impostata su PM. Infine si traccia la stringa `amPm` e se l'ora è maggiore o uguale a 12, viene visualizzato PM. In caso contrario, viene visualizzato AM.

Uso dell'istruzione if..else

L'istruzione condizionale `if..else` consente di provare una condizione e quindi eseguire un blocco di codice se la condizione è soddisfatta o un blocco di codice alternativo se la condizione non è soddisfatta.

Il codice seguente, ad esempio, prova se il valore di `x` è maggiore di 20, genera un'istruzione `trace()` in caso positivo o un'istruzione `trace()` diversa in caso negativo:

```
if (x > 20) {
    trace("x is > 20");
} else {
    trace("x is <= 20");
}
```

Se non si desidera eseguire un blocco di codice alternativo, è possibile utilizzare l'istruzione `if` senza l'istruzione `else`.

L'istruzione `if..else` in Flash è simile all'istruzione `if`. Se, ad esempio, si utilizza l'istruzione `if` per convalidare se il nome utente e la password immessi da un utente corrispondono a un valore contenuto in un database, potrebbe essere utile reindirizzare l'utente in base al risultato della verifica. Se il login è valido, l'utente può essere reindirizzato a una pagina di benvenuto tramite il blocco `if`. Se invece i dati immessi non sono validi, è possibile reindirizzare l'utente al form di login e visualizzare un messaggio di errore tramite il blocco `else`.

Per utilizzare un'istruzione if..else in un documento:

1. Selezionare File > Nuovo, quindi Documento Flash per creare un nuovo file FLA.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
// Crea una stringa che memorizza AM/PM in base all'ora del giorno
var amPm:String;
// Nessun parametro viene passato a Date e quindi viene restituita la
// data e l'ora corrente
var current_date:Date = new Date();
// Se l'ora corrente è maggiore o uguale a 12, la stringa amPm viene
// impostata su "PM"
if (current_date.getHours() >= 12) {
    amPm = "PM";
} else {
    amPm = "AM";
}
trace(amPm);
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

In questo codice si crea una stringa che memorizza AM o PM in base all'ora del giorno. Se l'ora corrente è maggiore o uguale a 12, la stringa `amPm` viene impostata su PM. Infine si traccia la stringa `amPm` e se l'ora è maggiore o uguale a 12, viene visualizzato PM. In caso contrario, nel pannello Output viene visualizzato AM.

Uso dell'istruzione if..else if

L'istruzione condizionale `if..else if` permette di provare più di una condizione. La sintassi è la seguente:

```
// Istruzione else-if
if (condition) {
    // Istruzioni;
} else if (condition) {
    // Istruzioni;
} else {
    // Istruzioni;
}
```

Un blocco `if..else if` viene utilizzato nei progetti Flash quando è necessario controllare una serie di condizioni. Se, ad esempio, si desidera visualizzare un'immagine diversa in base all'ora del giorno in cui l'utente esegue la visita, è possibile creare una serie di istruzioni `if` che determinano se è mattino, pomeriggio, sera o notte e quindi visualizzare la grafica adeguata.

Il codice seguente non solo controlla se il valore di x è maggiore di 20, ma anche se è negativo:

```
if (x > 20) {  
    trace("x is > 20");  
} else if (x < 0) {  
    trace("x is negative");  
}
```

Per utilizzare un'istruzione if..else if in un documento:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var now_date:Date = new Date();  
var currentHour:Number = now_date.getHours();  
// if l'ora corrente è precedente alle 11  
if (currentHour < 11) {  
    trace("Good morning");  
    // else..if l'ora corrente è precedente alle 15  
} else if (currentHour < 15) {  
    trace("Good afternoon");  
    // else..if l'ora corrente è precedente alle 20  
} else if (currentHour < 20) {  
    trace("Good evening");  
    // else l'ora corrente è tra le 20 e le 23.59  
} else {  
    trace("Good night");  
}
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

In questo codice si crea una stringa denominata `currentHour` che contiene il valore dell'ora corrente, ad esempio se sono le 18.19, `currentHour` contiene il numero 18. Per ottenere l'ora corrente viene utilizzato il metodo `getHours()` della classe `Date`.

L'istruzione `if..else if` può quindi essere utilizzata per tracciare le informazioni e visualizzarle nel pannello Output in base al numero restituito. Per ulteriori informazioni, vedere i commenti nel frammento di codice precedente.

Uso di un'istruzione switch

L'istruzione `switch` crea una struttura ad albero per le istruzioni ActionScript. Come nel caso dell'istruzione `if`, l'istruzione `switch` prova una condizione ed esegue le istruzioni se la condizione restituisce il valore `true`.

Quando viene utilizzata un'istruzione `switch`, l'istruzione `break` indica a Flash di ignorare il resto delle istruzioni in quel blocco `case` e passare alla prima istruzione che segue l'istruzione `switch` che le racchiude. Se un blocco `case` non contiene un'istruzione `break`, si verifica una condizione che viene detta "fall through", ovvero l'esecuzione non viene interrotta. In questa situazione, viene eseguita anche l'istruzione `case` seguente fino al raggiungimento di un'istruzione `break` o fino a quando l'istruzione `switch` termina. Questo comportamento è dimostrato nell'esempio seguente in cui la prima istruzione `case` non contiene un'istruzione `break` e quindi vengono eseguiti entrambi i blocchi di codice per i primi due blocchi `case` (A e B).

Tutte le istruzioni `switch` dovrebbero comprendere un'etichetta `case default` che deve sempre essere l'ultima etichetta `case` di un'istruzione `switch` e includere un'istruzione `break` che consenta di uscire dall'istruzione `switch` quando viene soddisfatta la condizione. Se, ad esempio, la condizione presente nell'esempio seguente restituisce A, vengono eseguite le istruzioni relative sia a `case A` che a `case B`, perché per `case A` non è presente un'istruzione `break`. Un'etichetta `case` che non consente di uscire dall'istruzione non include un'istruzione `break`, ma un commento al posto dell'istruzione `break`, come illustrato nell'esempio seguente dopo `case A`. Per la scrittura di istruzioni `switch`, utilizzare il formato seguente:

```
switch (condition) {  
    case A :  
        // istruzioni  
        // Continua la valutazione  
    case B :  
        // istruzioni  
        break;  
    case Z :  
        // istruzioni  
        break;  
    default :  
        // istruzioni  
        break;  
}
```

Per utilizzare un'istruzione switch in un documento:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
    // Utilizza il metodo String.fromCharCode() per restituire una
    // stringa.
    switch (String.fromCharCode(Key.getAscii())) {
        case "A" :
            trace("you pressed A");
            break;
        case "a" :
            trace("you pressed a");
            break;
        case "E" :
        case "e" :
            /* E non dispone di un'istruzione break, pertanto il blocco viene
            eseguito se si preme e o E. */
            trace("you pressed E or e");
            break;
        case "I" :
        case "i" :
            trace("you pressed I or i");
            break;
        default :
            /* Se il tasto premuto non viene rilevato da uno dei case
            precedenti, viene eseguito questo case predefinito. */
            trace("you pressed some other key");
    }
};
Key.addListener(listenerObj);
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

Digitare alcune lettere con la tastiera, incluse le lettere a, e o i. Quando vengono digitate queste tre lettere, vengono visualizzate le istruzioni `trace` del precedente codice ActionScript. La riga di codice crea un nuovo oggetto da utilizzare come listener della classe `Key` per notificare all'evento `onKeyDown()` quando l'utente preme un tasto. Il metodo `Key.getAscii()` restituisce il codice ASCII dell'ultimo tasto che l'utente preme o rilascia. È quindi necessario utilizzare il metodo `String.fromCharCode()` per restituire una stringa che contenga i caratteri rappresentati dai valori ASCII nei parametri. Dato che "E" non dispone di un'istruzione `break`, il blocco viene eseguito se l'utente preme il tasto *e* o *E*. Se l'utente preme un tasto che non viene rilevato da uno dei tre primi `case`, viene eseguito il `case predefinito`.

Uso delle istruzioni try..catch e try..catch..finally

L'uso del blocco `try..catch..finally` consente di aggiungere la gestione degli errori alle applicazioni Flash. Con le parole chiave `try..catch..finally` è possibile racchiudere un blocco di codice in cui si può verificare un errore e rispondere all'errore. Se una porzione di codice all'interno del blocco `try` genera un errore (utilizzando l'istruzione `throw`), il controllo passa al blocco `catch`, se presente e quindi al blocco `finally`, se presente. Il blocco `finally` opzionale viene sempre eseguito, indipendentemente dal fatto che sia stato generato un errore o meno.

Se il codice nel blocco `try` non genera un errore (ovvero se il blocco `try` viene completato normalmente), il codice nel blocco `finally` viene comunque eseguito.

NOTA

Il blocco `finally` viene eseguito anche se il blocco `try` viene chiuso mediante un'istruzione `return`

Il formato delle istruzioni `try..catch` e `try..catch..finally` è il seguente:

```
// Try-catch
try {
    // istruzioni
} catch (myError) {
    // istruzioni
}

// Try-catch-finally
try {
    // istruzioni
} catch (myError) {
    // istruzioni
} finally {
    // istruzioni
}
```

Ogni volta che il codice genera un errore, è possibile scrivere gestori personalizzati per gestire l'errore ed eseguire operazioni appropriate. Potrebbe essere necessario caricare dati esterni da un servizio Web o un file di testo o visualizzare un messaggio di errore all'utente finale. È possibile inoltre utilizzare il blocco `catch` per tentare il collegamento a un servizio Web che avverte un amministratore del verificarsi di un determinato errore affinché possa garantire il corretto funzionamento dell'applicazione.

Per utilizzare il blocco try..catch..finally per la convalida dei dati prima di eseguire operazioni di divisione su alcuni numeri:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var n1:Number = 7;
var n2:Number = 0;
try {
    if (n2 == 0) {
        throw new Error("Unable to divide by zero");
    }
    trace(n1/n2);
} catch (err:Error) {
    trace("ERROR! " + err.toString());
} finally {
    delete n1;
    delete n2;
}
```

3. Selezionare Controllo > Prova filmato per provare il documento.
4. Nel pannello Output viene visualizzato Unable to divide by zero.
5. Tornare all'ambiente di creazione e modificare la seguente riga di codice:

```
var n2:Number = 0;
in
var n2:Number = 2;
```
6. Selezionare Controllo > Invio per provare nuovamente il documento.

Se il valore n2 è uguale a zero, viene generato un errore che viene rilevato dal blocco catch che, a sua volta, visualizza un messaggio nel pannello Output. Se il valore y non è uguale a zero, il pannello Output visualizza il risultato di n1 diviso per n2. Il blocco finally viene eseguito indipendentemente dal fatto che si verifichi un errore ed elimina i valori delle variabili n1 e n2 dal documento Flash.

Quando si verifica un errore è possibile anche non limitarsi a generare nuove istanze della classe Error, ma estendere la classe Error per creare errori personalizzati, come nell'esempio che segue.

Per creare un errore personalizzato:

1. Selezionare File > Nuovo e creare un nuovo file ActionScript.
2. Selezionare File > Salva con nome e assegnare al file il nome **DivideByZeroException.as**.
3. Immettere il codice ActionScript seguente nel pannello Script:

```
// In DivideByZeroException.as:  
class DivideByZeroException extends Error {  
    var message:String = "Divide By Zero error";  
}
```
4. Salvare il file ActionScript.
5. Creare un nuovo documento Flash denominato **exception_test.fla** nella stessa directory del file ActionScript e salvare il file.
6. Immettere il codice ActionScript seguente nel pannello Azioni del fotogramma 1 della linea temporale principale:

```
var n1:Number = 7;  
var n2:Number = 0;  
try {  
    if (n2 == 0) {  
        throw new DivideByZeroException();  
    } else if (n2 < 0) {  
        throw new Error("n2 cannot be less than zero");  
    } else {  
        trace(n1/n2);  
    }  
} catch (err:DivideByZeroException) {  
    trace(err.toString());  
} catch (err:Error) {  
    trace("An unknown error occurred; " + err.toString());  
}
```

7. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file nell'ambiente di prova.

Dato che il valore `n2` è uguale a 0, Flash genera la classe di errore personalizzata `DivideByZeroException` e visualizza `Divide By Zero error` nel pannello Output. Se si modifica il valore di `n2` nella riga due da 0 a -1 e si riprova il documento Flash, nel pannello Output viene visualizzato `An unknown error occurred; n2 cannot be less than zero`. Se si imposta il valore `n2` su un numero maggiore di 0, il risultato della divisione viene visualizzato nel pannello Output. Per ulteriori informazioni sulla creazione di classi personalizzate, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Informazioni sull'operatore condizionale e sintassi alternativa

È possibile anche utilizzare l'operatore condizionale (`? :`) per creare *espressioni condizionali*, convertendo istruzioni `if...else` semplici in una sola riga di codice e riducendo così la quantità di codice da scrivere per ottenere lo stesso risultato. Il codice ActionScript risulta tuttavia più difficile da leggere.

La condizione seguente, scritta senza operatore condizionale, verifica se la variabile `numTwo` è maggiore di zero e restituisce il risultato di `numOne/numTwo` o la stringa `carrot`:

```
var numOne:Number = 8;
var numTwo:Number = 5;
if (numTwo > 0) {
    trace(numOne / numTwo); // 1.6
} else {
    trace("carrot");
}
```

Con un'espressione condizionale, lo stesso codice avrebbe il seguente formato:

```
var numOne:Number = 8;
var numTwo:Number = 0;
trace((numTwo > 0) ? numOne/numTwo : "carrot");
```

In questo caso, la sintassi più breve riduce la leggibilità e pertanto non è preferibile. Se si devono utilizzare operatori condizionali, inserire la condizione iniziale (prima del punto di domanda `[?]`) all'interno di parentesi per migliorare la leggibilità del codice. Di seguito è riportato un esempio di codice ActionScript con leggibilità migliore:

```
var numOne:Number;
(numOne >= 5) ? numOne : -numOne;
```

È possibile scrivere un'istruzione condizionale che restituisca un valore booleano, come illustrato nell'esempio seguente:

```
if (cartArr.length > 0) {
    return true;
} else {
    return false;
}
```

Tuttavia, il codice ActionScript seguente è preferibile rispetto a quello riportato in precedenza:

```
return (cartArr.length > 0);
```

Il secondo frammento di codice è più breve e il numero di espressioni da valutare è inferiore; È inoltre più semplice da leggere e comprendere.

Quando si scrivono condizioni complesse, utilizzare le parentesi tonde `()` per raggruppare le condizioni. Se non si utilizzano le parentesi, si potrebbero verificare errori di priorità degli operatori. Per ulteriori informazioni sulla priorità degli operatori, consultare il [“Informazioni sulla priorità e l'associatività degli operatori” a pagina 146](#).

Nel codice seguente, ad esempio, non vengono utilizzate le parentesi per la condizione:

```
if (fruit == "apple" && veggie == "leek") {}
```

Nel codice seguente la sintassi è corretta in quanto vengono aggiunte le parentesi per le condizioni:

```
if ((fruit == "apple") && (veggie == "leek")) {}
```

Come ripetere azioni utilizzando i cicli

ActionScript può ripetere un'azione per un determinato numero di volte o quando una condizione specifica risulta soddisfatta. Con i cicli è possibile ripetere una serie di istruzioni quando una determinata condizione è `true`. In ActionScript esistono quattro tipi di cicli: cicli `for`, cicli `for..in`, cicli `while` e cicli `do..while`. Ogni tipo di ciclo si comporta in modo diverso e serve a scopi diversi.

Nella maggior parte dei cicli, un contatore verifica il numero di volte per cui un ciclo viene eseguito. Ogni ripetizione di un ciclo è denominata *iterazione*. È possibile dichiarare una variabile, quindi scrivere un'istruzione che ne incrementa o decrementa il valore a ogni esecuzione del ciclo. Nell'azione `for`, il contatore e l'istruzione che lo incrementa sono parti integranti dell'azione.

Ciclo	Descrizione
cicli <code>for</code>	Ripete un'azione utilizzando un contatore incorporato.
cicli <code>for..in</code>	Esegue un ciclo tra gli elementi secondari di un clip filmato o di un oggetto.
cicli <code>while</code>	Ripete un'azione per tutto il tempo in cui una condizione esiste.
cicli <code>do..while</code>	Simile al ciclo <code>while</code> , ad eccezione del fatto che l'espressione viene valutata alla fine del blocco di codice, affinché il ciclo venga sempre eseguito almeno una volta.

Il più comune è il ciclo `for` che esegue un'iterazione su un blocco di codice per un numero di volte predefinito. Se, ad esempio, è presente un array di elementi e si desidera eseguire una serie di istruzioni su ogni elemento dell'array, si utilizza il ciclo `for` e si esegue l'iterazione da 0 al numero di elementi nell'array. Un altro tipo di ciclo è il ciclo `for..in`, che può essere molto utile quando si desidera eseguire un'iterazione di ogni coppia nome/valore all'interno di un oggetto e quindi eseguire un tipo di azione, ad esempio se si esegue il debug di progetti Flash e si desidera visualizzare i valori caricati da origini esterne, quali servizi Web o file di testo/XML esterni. I due ultimi cicli (`while` e `do..while`) sono utili per eseguire iterazioni su una serie di istruzioni, se non si conosce a priori il numero di iterazioni da eseguire. In tal caso è possibile utilizzare un ciclo `while` per eseguire iterazioni per tutto il tempo in cui una determinata condizione è soddisfatta.

ActionScript può ripetere un'azione per un determinato numero di volte o quando una condizione specifica risulta soddisfatta. Per creare cicli, utilizzare le azioni `while`, `do..while`, `for` e `for..in`. Questa sezione contiene informazioni generali sui cicli. Per ulteriori informazioni su ognuno dei cicli, vedere le seguenti procedure.

Per ripetere un'azione finché una condizione non viene soddisfatta:

- Usare l'istruzione `while`.

Un ciclo `while` valuta un'espressione `e`, se l'espressione risulta `true`, esegue il codice contenuto nel corpo del ciclo. Dopo l'esecuzione di tutte le istruzioni nel corpo, l'espressione viene nuovamente valutata. Nell'esempio seguente il ciclo viene eseguito quattro volte:

```
var i:Number = 4;
while (i>0) {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
}
```

È possibile utilizzare l'istruzione `do..while` per creare un tipo di ciclo analogo al ciclo `while`. In un ciclo `do...while`, l'espressione viene valutata alla fine del blocco di codice, affinché il ciclo venga sempre eseguito almeno una volta.

Questo comportamento viene illustrato nel seguente esempio:

```
var i:Number = 4;
do {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
} while (i>0);
```

Per ulteriori informazioni sull'istruzione `while`, vedere “[Uso dei cicli while](#)” a pagina 126.

Per ripetere un'azione tramite un contatore incorporato:

- Usare l'istruzione `for`.

Nella maggior parte dei cicli, un contatore verifica il numero di volte per cui un ciclo viene eseguito. Ogni ripetizione di un ciclo è denominata *iterazione*. È possibile dichiarare una variabile, quindi scrivere un'istruzione che ne incrementa o decrementa il valore a ogni esecuzione del ciclo. Nell'azione `for`, il contatore e l'istruzione che lo incrementa sono parti integranti dell'azione.

Nell'esempio seguente la prima espressione (`var i:Number = 4`) è l'espressione iniziale che viene valutata prima della prima iterazione. La seconda espressione (`i > 0`) è la condizione che viene verificata ogni volta prima dell'esecuzione del ciclo. La terza espressione (`i--`) è denominata *post espressione* e viene valutata dopo ogni esecuzione del ciclo.

```
for (var i:Number = 4; i > 0; i--) {  
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});  
}
```

Per ulteriori informazioni sull'istruzione `for`, vedere “[Uso dei cicli for](#)” a pagina 123.

Per eseguire un ciclo tra gli elementi secondari di un clip filmato o di un oggetto:

- Usare l'istruzione `for...in`.

Gli elementi secondari sono altri clip filmato, funzioni, oggetti e variabili. Nell'esempio seguente, l'istruzione `trace` viene utilizzata per inviare i risultati al pannello Output:

```
var myObject:Object = {name:'Joe', age:25, city:'San Francisco'};  
var propertyName:String;  
for (propertyName in myObject) {  
    trace("myObject has the property: " + propertyName + ", with the  
    value: " + myObject[propertyName]);  
}
```

Questo esempio produce i seguenti risultati nel pannello Output:

```
myObject has the property: name, with the value: Joe  
myObject has the property: age, with the value: 25  
myObject has the property: city, with the value: San Francisco
```

Se si desidera che lo script esegua un'iterazione su un determinato tipo di elemento secondario, ad esempio soltanto su clip filmato secondari, utilizzare `for...in` con l'operatore `typeof`. Nell'esempio seguente, un'istanza del clip filmato secondario (chiamata `instance2`) si trova all'interno di un'istanza clip filmato sullo Stage.

Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
for (var myName in this) {  
    if (typeof (this[myName]) == "movieclip") {  
        trace("Nome oggetto secondario clip filmato " + myName);  
    }  
}
```

Per ulteriori informazioni sull'istruzione `for...in`, vedere “[Uso dei cicli for..in](#)” a [pagina 125](#).

AVVISO

Le iterazioni sono eseguite in Flash molto velocemente da Flash Player, mentre i cicli dipendono notevolmente dal tipo di processore. Maggiore è il numero di iterazioni di un ciclo, maggiore è il numero di istruzioni eseguite all'interno di ogni blocco, maggiore è la quantità di risorse del processore consumate. Cicli non ottimizzati possono provocare problemi di prestazioni e di stabilità.

Per ulteriori informazioni su ogni istruzione, vedere le singole sezioni successive a questo capitolo, ad esempio “[Uso dei cicli while](#)” a [pagina 126](#) e le relative voci nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sulla creazione di cicli e l'uscita dai cicli

L'esempio seguente mostra un array semplice di nomi di mesi. Un ciclo `for` esegue un'iterazione da 0 al numero di elementi nell'array e visualizza ogni elemento nel pannello Output:

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",  
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");  
var i:Number;  
for (i = 0; i < monthArr.length; i++) {  
    trace(monthArr[i]);  
}
```

Quando si lavora con array, semplici o complessi, è necessario avere familiarità con una condizione detta *ciclo infinito*. Come suggerisce il nome, un ciclo infinito è un ciclo senza condizione di fine. Questa situazione può causare problemi, ad esempio il blocco dell'applicazione Flash o la mancata risposta di un documento Flash in un browser Web o ancora un comportamento molto incoerente del documento Flash. Di seguito è riportato un esempio di un codice con un ciclo infinito:

```
// CODICE SCORRETTO: crea un ciclo infinito
// Può causare errori
var i:Number;
for (i = 0; i < 10; i--) {
    trace(i);
}
```

Il valore di *i* viene inizializzato su 0 e la condizione finale è soddisfatta quando *i* è maggiore o uguale a 10; dopo ogni iterazione il valore di *i* viene incrementato. L'errore è probabilmente visibile subito: se il valore di *i* viene ridotto dopo ogni iterazione del ciclo, la condizione finale non viene mai soddisfatta. I risultati variano in base al computer su cui il codice viene eseguito e la velocità di raggiungimento dell'errore dipende dalla velocità della CPU e da altri fattori. Su un dato computer, ad esempio, il ciclo può essere eseguito circa 142.620 volte prima di visualizzare un messaggio di errore.

In una finestra di dialogo viene visualizzato il messaggio di errore seguente:

A script in this movie is causing Flash Player to run slowly. If it continues to run, your computer may become unresponsive. Do you want to abort the script?

Quando si lavora con un ciclo, in particolare con i cicli *while* e *do..while*, verificare sempre che sia possibile uscire correttamente dal ciclo e non si crei un ciclo infinito.

Per ulteriori informazioni sul controllo dei cicli, vedere “[Uso di un'istruzione switch](#)” a pagina 113.

Uso dei cicli *for*

Il ciclo *for* permette di eseguire iterazioni su una variabile per verificare un intervallo di valori specifico. È utile quando si conosce esattamente il numero di volte per cui è necessario ripetere una serie di istruzioni ActionScript, ad esempio se si desidera creare un determinato numero di duplicati di un clip filmato sullo stage o eseguire iterazioni su un array ed effettuare un'operazione su ogni elemento dell'array. Un ciclo *for* ripete un'azione utilizzando un contatore incorporato. Il contatore e l'istruzione che lo incrementa sono parti integranti dell'istruzione *for*. L'istruzione *for* ha il seguente formato di base:

```
for (init; condition; update) {
    // Istruzioni;
}
```

A un'istruzione `for` è necessario fornire tre espressioni: una variabile impostata su un valore iniziale, un'istruzione condizionale che determina quando il ciclo termina e un'espressione che cambia il valore della variabile a ogni ciclo. Il codice seguente, ad esempio, esegue cinque iterazioni. Il valore della variabile `i` inizia a 0 e termina a 4 e l'output è rappresentato dai numeri da 0 a 4, ognuno su una riga separata.

```
var i:Number;  
for (i = 0; i < 5; i++) {  
    trace(i);  
}
```

Nell'esempio seguente la prima espressione (`i = 0`) è l'espressione iniziale che viene valutata prima della prima iterazione. La seconda espressione (`i < 5`) è la condizione che viene verificata prima di ogni esecuzione del ciclo. La terza espressione (`i++`) è denominata *post espressione* e viene valutata dopo ogni esecuzione del ciclo.

Per creare un ciclo `for`:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Creare un clip filmato sullo stage.
3. Nel pannello Libreria, fare clic con il pulsante destro del mouse sul simbolo del clip filmato e selezionare Concatenamento dal menu di scelta rapida.
4. Selezionare la casella di controllo Esporta per ActionScript e digitare `libraryLinkageClassName` nel campo di testo di input Classe. Fare clic su OK.
5. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var i:Number;  
for (i = 0; i < 5; i++) {  
    this.attachMovie("libraryLinkageClassName", "clip" + i + "_mc", i,  
    {_x:(i * 100)});  
}
```

6. Selezionare Controllo > Prova filmato per provare il codice in Flash Player.

Si noti che nella parte superiore dello stage sono stati duplicati cinque clip filmato. ActionScript duplica il simbolo del clip filmato nella libreria e riposiziona i clip sullo stage alle coordinate `x` di 0, 100, 200, 300 e 400 pixel. Il ciclo viene eseguito cinque volte e alla variabile `i` viene assegnato un valore da 0 a 4. All'ultima iterazione del ciclo, il valore di `i` viene incrementato a 4 e la seconda espressione (`i < 5`) non è più soddisfatta, pertanto il ciclo termina.

Inserire uno spazio dopo ogni espressione all'interno di un'istruzione `for`. Per ulteriori informazioni, vedere `for` statement nella *Guida di riferimento di ActionScript 2.0*.

Uso dei cicli for..in

Un ciclo `for..in` consente di eseguire *iterazioni* tra gli elementi secondari di un clip filmato, le proprietà di un oggetto o gli elementi di un array. Gli elementi secondari, a cui si è fatto riferimento in precedenza, sono altri clip filmato, funzioni, oggetti e variabili. Gli utilizzi comuni del ciclo `for..in` comprendono l'iterazione su istanze di una linea temporale o su coppie valore/chiave all'interno di un oggetto. L'iterazione su oggetti può risultare efficace per eseguire il debug di applicazioni in quanto vengono visualizzati i valori restituiti dai servizi Web o da documenti esterni quali file di testo o XML.

È possibile, ad esempio, ricorrere a un ciclo `for...in` per eseguire iterazioni sulle proprietà di un oggetto generico. Le proprietà degli oggetti non vengono ordinate in base a criteri particolari, ma inserite in ordine imprevedibile:

```
var myObj:Object = {x:20, y:30};  
for (var i:String in myObj) {  
    trace(i + ": " + myObj[i]);  
}
```

Il seguente output del codice viene visualizzato nel pannello Output:

```
x: 20  
y: 30
```

È possibile anche eseguire iterazioni sugli elementi di un array:

```
var myArray:Array = ["one", "two", "three"];  
for (var i:String in myArray) {  
    trace(myArray[i]);  
}
```

Il seguente output del codice viene visualizzato nel pannello Output:

```
tre  
due  
uno
```

Per ulteriori informazioni su oggetti e proprietà, consultare “[Tipo di dati oggetto](#)” [a pagina 341](#).

NOTA

Non è possibile eseguire iterazioni sulle proprietà degli oggetti che sono istanze di classi personalizzate se la classe non è dinamica. Anche in quest'ultimo caso, è possibile solo eseguire iterazioni sulle proprietà aggiunte in modo dinamico.

NOTA

Le parentesi graffe (`()`) utilizzate per racchiudere il blocco di istruzioni che devono essere eseguite dall'istruzione `for..in` non sono necessarie se viene eseguita una sola istruzione.

L'esempio seguente utilizza `for..in` per eseguire un'iterazione sulle proprietà di un oggetto:

Per creare un ciclo for:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var myObj:Object = {name:"Tara", age:27, city:"San Francisco"};
var i:String;
for (i in myObj) {
    trace("myObj." + i + " = " + myObj[i]);
}
```

3. Selezionare Controllo > Prova filmato per provare il codice in Flash Player.

Quando si prova il file SWF, viene visualizzato il testo seguente nel pannello Output:

```
myObj.name = Tara
myObj.age = 27
myObj.city = San Francisco
```

Se si scrive un ciclo `for..in` in un file di classe (un file ActionScript esterno), i membri dell'istanza non sono disponibili all'interno del ciclo, ma i membri statici sì. Se invece si scrive un ciclo `for..in` in un file FLA per un'istanza della classe, i membri dell'istanza sono disponibili, ma i membri statici no. Per ulteriori informazioni sulla scrittura di file di classe, consultare il [Capitolo 6, “Classi” a pagina 197](#). Per ulteriori informazioni, vedere `for..in` statement nella *Guida di riferimento di ActionScript 2.0*.

Uso dei cicli while

Utilizzare l'istruzione `while` per ripetere un'azione mentre una condizione è soddisfatta, analogamente a un'istruzione `if` che viene ripetuta per tutto il tempo che la condizione rimane `true`.

Un ciclo `while` valuta un'espressione `e`, se l'espressione risulta `true`, esegue il codice contenuto nel corpo del ciclo. Se la condizione restituisce `true`, viene eseguita un'istruzione o una serie di istruzioni e quindi si torna a valutare la condizione. Dopo che la condizione ha restituito `false`, l'istruzione o la serie di istruzioni viene ignorata e il ciclo termina. I cicli `while` possono essere molto utili se non si conosce il numero di iterazioni necessarie su un blocco di codice.

Il codice seguente, ad esempio, traccia e visualizza numeri nel pannello Output:

```
var i:Number = 0;
while (i < 5) {
    trace(i);
    i++;
}
```

I numeri seguenti vengono tracciati e visualizzati nel pannello Output:

```
0  
1  
2  
3  
4
```

Il ciclo `while` presenta uno svantaggio rispetto al ciclo `for`: è più facile scrivere cicli infiniti. A differenza del ciclo `while`, l'esempio di codice del ciclo `for` non viene compilato se si omette l'espressione che incrementa la variabile del contatore. Senza l'espressione che incrementa `i`, il ciclo diventa infinito.

Per creare e utilizzare un ciclo `while` in un file FLA, seguire l'esempio indicato di seguito.

Per creare un ciclo `while`:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Aprire il pannello Componenti e trascinare un componente DataSet sullo stage.
3. Aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà) e digitare il nome di istanza `users_ds`.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var users_ds:mx.data.components.DataSet;  
//  
users_ds.addItem({name:"Irving", age:34});  
users_ds.addItem({name:"Christopher", age:48});  
users_ds.addItem({name:"Walter", age:23});  
//  
users_ds.first();  
while (users_ds.hasNext()) {  
    trace("name:" + users_ds.currentItem["name"] + ", age:" +  
        users_ds.currentItem["age"]);  
    users_ds.next();  
}
```

5. Selezionare Controllo > Prova filmato per provare il documento.

Le informazioni seguenti vengono visualizzate nel pannello Output:

```
name:Irving, age:34  
name:Christopher, age:48  
name:Walter, age:23
```

Per ulteriori informazioni, vedere `while` statement nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sui cicli do..while

È possibile usare l'istruzione `do...while` per creare un tipo di ciclo analogo al ciclo `while`. Tuttavia in un ciclo `do...while`, l'espressione viene valutata alla fine del blocco di codice, ovvero dopo la sua esecuzione, affinché il ciclo venga sempre eseguito almeno una volta. Le istruzioni vengono eseguite solo se la condizione restituisce `true`.

Il codice seguente mostra un esempio semplice di un ciclo `do..while` che genera output anche se la condizione non è soddisfatta.

```
var i:Number = 5;
do {
    trace(i);
    i++;
} while (i < 5);
// Output: 5
```

Se si utilizzano cicli, evitare che si creino cicli infiniti. Se la condizione di un ciclo `do..while` restituisce sempre `true`, si crea un ciclo infinito che causa la visualizzazione di un avviso o il blocco di Flash Player. Se si conosce il numero di iterazioni da eseguire, utilizzare un ciclo `for`. Per ulteriori informazioni ed esempi dei cicli `do..while` statement, vedere la *Guida di riferimento di ActionScript 2.0*.

Uso di cicli nidificati in ActionScript

L'esempio seguente dimostra come creare un array di oggetti e visualizzare ogni valore della struttura nidificata, utilizzando il ciclo `for` per eseguire iterazioni su ogni elemento dell'array e il ciclo `for..in` per eseguire iterazioni su ogni coppia chiave/valore degli oggetti nidificati.

Per nidificare un ciclo all'interno di un altro ciclo:

1. Creare un nuovo documento Flash.
2. Selezionare File > Salva con nome e assegnare al documento il nome **loops.fla**.
3. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var myArr:Array = new Array();
myArr[0] = {name:"One", value:1};
myArr[1] = {name:"Two", value:2};
//
var i:Number;
var item:String;
for (i = 0; i < myArr.length; i++) {
    trace(i);
    for (item in myArr[i]) {
        trace(item + ": " + myArr[i][item]);
    }
    trace("");
}
```

4. Selezionare Controllo > Prova filmato per provare il codice.

Le informazioni seguenti vengono visualizzate nel pannello Output:

```
0
name: One
value: 1
```

```
1
name: Two
value: 2
```

Dato che si conosce il numero di elementi dell'array, è possibile eseguire iterazioni su ogni elemento tramite un ciclo `for` semplice. Ogni elemento nell'array può avere coppie nome/valore diverse, pertanto è possibile utilizzare un ciclo `for...in` per eseguire iterazioni su ogni valore e visualizzare i risultati nel pannello Output.

Informazioni sugli array

Un *array* è un oggetto le cui proprietà sono identificate da un numero che ne rappresenta la posizione nella struttura dell'array. Un array è fondamentalmente un elenco di elementi. È importante tenere a mente che non occorre che gli elementi dell'array abbiano lo stesso tipo di dati. È possibile inserire numeri, dati, stringhe, oggetti e anche aggiungere un array nidificato per ogni indice di array.

L'esempio seguente mostra un array semplice di nomi di mesi.

```
var myArr:Array = new Array();
myArr[0] = "January";
myArr[1] = "February";
myArr[2] = "March";
myArr[3] = "April";
```

L'array precedente può essere riscritto come segue:

```
var myArr:Array = new Array("January", "February", "March", "April");
```

In alternativa, è possibile utilizzare la sintassi abbreviata:

```
var myArr:Array = ["January", "February", "March", "April"];
```

Un array è come una struttura per i dati e può essere paragonato a un edificio ad uso uffici, dove ogni piano contiene dati diversi, ad esempio la contabilità (*accounting*) al terzo piano e la progettazione (*engineering*) al quinto piano. È possibile memorizzare diversi tipi di dati in un solo array, compresi altri array. Ogni piano dell'edificio può contenere più tipi di contenuto, ad esempio la direzione (*executives*) e la contabilità (*accounting*) possono trovarsi entrambi al terzo piano.

Un array contiene *elementi* che equivalgono a ogni piano dell'edificio. Ogni elemento ha una posizione numerica (l'*indice*) che corrisponde alla modalità con cui si fa riferimento alla posizione di ogni elemento dell'array. Analogamente, ogni piano dell'edificio ha un numero. Ogni elemento può contenere dati (vale a dire un numero, una stringa, un valore booleano o anche un array o un oggetto) oppure essere vuoto.

L'array può inoltre essere controllato e modificato. Potrebbe ad esempio essere necessario spostare il reparto progettazione al piano terra dell'edificio. È possibile spostare i valori in punti diversi dell'array e modificare le dimensioni dell'array, come è possibile aggiungere o eliminare piani all'edificio. A tale scopo, gli elementi vengono aggiunti o rimossi e i valori vengono spostati in elementi diversi.

L'edificio (l'array) contiene pertanto piani (gli elementi), ogni piano è numerato (l'indice) e contiene uno o più reparti (i valori).

Per ulteriori informazioni sulla modifica di array, vedere “[Informazioni sulla modifica di array](#)” a pagina 132. Per ulteriori informazioni sull'uso di array e sugli indici, vedere “[Uso degli array](#)” a pagina 130. Per informazioni sull'aggiunta e la rimozione di elementi, vedere “[Informazioni su aggiunta e rimozione di elementi](#)” a pagina 134. Per informazioni sull'operatore di accesso agli array, vedere “[Uso dell'operatore punto e dell'operatore di accesso agli array](#)” a pagina 151.

È possibile trovare un file di origine di esempio, array.fla, nella cartella Samples sul disco rigido. L'esempio illustra la gestione di array con ActionScript. Il codice dell'esempio crea un array; quindi ordina, aggiunge e rimuove le voci di due componenti List. Il file di esempio si trova nelle seguenti cartelle:

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/Arrays*.

Uso degli array

Gli array possono essere utilizzati in modi diversi, ad esempio per memorizzare elenchi di oggetti quali un elenco di elementi restituiti. Se si caricano dati da server Web remoti, i dati potrebbero essere ricevuti sotto forma di array di oggetti nidificati. Spesso gli array contengono dati di formato simile, ad esempio se si crea un'applicazione audio in Flash, la sequenza di brani di un utente potrebbe essere memorizzata come array di informazioni sui brani, a loro volta memorizzate in oggetti. Ogni oggetto contiene il nome del brano, dell'artista, la durata del brano, il percorso di un file audio (ad esempio MP3) o altre informazioni che si desidera associare a un determinato file.

La posizione di un elemento nell'array è detta *indice*. Tutti gli array sono con base zero, ovvero [0] è il primo elemento dell'array, [1] è il secondo, e così via.

Esistono diversi tipi di array, come illustrato nelle sezioni seguenti. Gli array più comuni utilizzano un indice numerico per la ricerca di un determinato elemento in un *array indicizzato*. Il secondo tipo di array è detto *array associativo* e utilizza un indice di testo anziché un indice numerico per la ricerca di informazioni. Per ulteriori informazioni sugli array comuni, vedere “[Informazioni sugli array](#)” a pagina 129. Per ulteriori informazioni sugli array associativi, vedere “[Creazione di array associativi](#)” a pagina 139. Per ulteriori informazioni sugli array multidimensionali, vedere “[Creazione di array multidimensionali](#)” a pagina 136. Per informazioni sull'operatore di accesso agli array, vedere “[Uso dell'operatore punto e dell'operatore di accesso agli array](#)” a pagina 151.

La classe incorporata Array consente di accedere agli array e di manipolarli. Per creare un oggetto Array, utilizzare la funzione di costruzione `new Array()` o l'operatore di accesso agli array (`[]`) che permette anche di accedere agli elementi di un array. Nell'esempio seguente viene utilizzato un array indicizzato.

Per utilizzare array nel codice:

1. Creare un nuovo documento Flash e salvarlo come **basicArrays.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Definisce un nuovo array
var myArr:Array = new Array();
// Definisce valori su due indici
myArr[1] = "value1";
myArr[0] = "value0";
// Esegue un'iterazione sugli elementi dell'array
var i:String;
for (i in myArr) {
    // Traccia le coppie chiave/valore
    trace("key: " + i + ", value: " + myArr[i]);
}
```

Nella prima riga del codice ActionScript viene definito un nuovo array per contenere i valori, quindi si definiscono dati (`value0` e `value1`) su due indici dell'array. Si utilizza un ciclo `for...in` per eseguire iterazioni su ciascun elemento dell'array e visualizzare le coppie chiave/valore nel pannello Output utilizzando un'istruzione `trace`.

3. Selezionare Controllo > Prova filmato per provare il codice.

Il testo seguente viene visualizzato nel pannello Output:

```
key: 0, value: value0
key: 1, value: value1
```

Per ulteriori informazioni sui cicli `for...in`, vedere “[Uso dei cicli for..in](#)” a pagina 125.

Per informazioni su come creare diversi tipi di array, consultare le sezioni seguenti:

- “Creazione di array indicizzati” a pagina 135
- “Creazione di array multidimensionali” a pagina 136
- “Creazione di array associativi” a pagina 139

È possibile trovare un file di origine di esempio, array.fla, nella cartella Samples sul disco rigido. L'esempio illustra la gestione di array con ActionScript. Il codice dell'esempio crea un array; quindi ordina, aggiunge e rimuove le voci di due componenti List. Il file di esempio si trova nelle seguenti cartelle:

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/Arrays*.

Informazioni sulla modifica di array

L'array può essere controllato e modificato tramite ActionScript. È possibile spostare valori all'interno dell'array o modificarne la dimensione. Se ad esempio si desidera scambiare i dati di due indici di un array, è possibile utilizzare il seguente codice:

```
var buildingArr:Array = new Array();
buildingArr[2] = "Accounting";
buildingArr[4] = "Engineering";
trace(buildingArr); // undefined,undefined,Accounting,undefined,Engineering

var temp_item:String = buildingArr[2];
buildingArr[2] = buildingArr[4];
buildingArr[4] = temp_item;
trace(buildingArr); // undefined,undefined,Engineering,undefined,Accounting
```

Nell'esempio precedente è necessario creare una variabile temporanea per il motivo illustrato di seguito. Se il contenuto dell'indice 4 dell'array venisse copiato nell'indice 2 dell'array e vice versa, il contenuto originale dell'indice 2 andrebbe perso. Copiando il valore di un indice dell'array in una variabile temporanea, è possibile salvare il valore e copiarlo nuovamente in seguito nel codice. Se, ad esempio, si utilizzasse il codice seguente, il valore dell'indice 2 dell'array (Accounting) andrebbe perso. Si avrebbero due team di progettazione, ma nessun reparto contabilità.

```
// Modalità scorretta; non viene utilizzata la variabile temporanea
buildingArr[2] = buildingArr[4];
buildingArr[4] = buildingArr[2];
trace(buildingArr);
// undefined,undefined,Engineering,undefined,Engineering
```

È possibile trovare un file di origine di esempio, array.fla, nella cartella Samples sul disco rigido. L'esempio illustra la gestione di array con ActionScript. Il codice dell'esempio crea un array; quindi ordina, aggiunge e rimuove le voci di due componenti List. Il file di esempio si trova nelle seguenti cartelle:

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/Arrays*.

Informazioni sul riferimento alla lunghezza e su come determinarla

Quando si lavora con gli array, è spesso necessario conoscere il numero degli elementi contenuti in un array, in particolare se si scrivono cicli `for` che eseguono iterazioni su ogni elemento dell'array ed eseguono una serie di istruzioni. Il frammento di codice seguente contiene un esempio:

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
trace(monthArr); // Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
var i:Number;
for (i = 0; i < monthArr.length; i++) {
    monthArr[i] = monthArr[i].toUpperCase();
}
trace(monthArr); // JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

Nell'esempio appena riportato, viene creato un array e compilato con nomi di mesi. Il contenuto e la lunghezza dell'array vengono visualizzati. Un ciclo `for` esegue un'iterazione su ogni elemento dell'array e converte il valore in maiuscole. Il contenuto dell'array viene quindi nuovamente visualizzato.

Nel codice ActionScript seguente, se si crea un elemento all'indice 5 dell'array, la lunghezza dell'array restituisce 6 (perché l'array ha base zero) e non il numero effettivo di elementi contenuti nell'array come previsto:

```
var myArr:Array = new Array();
myArr[5] = "five";
trace(myArr.length); // 6
trace(myArr); // undefined,undefined,undefined,undefined,undefined,5
```

Per ulteriori informazioni sui cicli `for`, vedere “[Uso dei cicli for](#)” a pagina 123. Per informazioni sull'operatore di accesso agli array, vedere “[Uso dell'operatore punto e dell'operatore di accesso agli array](#)” a pagina 151.

È possibile trovare un file di origine di esempio, array.fla, nella cartella Samples sul disco rigido. L'esempio illustra la gestione di array con ActionScript. Il codice dell'esempio crea un array; quindi ordina, aggiunge e rimuove le voci di due componenti List. Il file di esempio si trova nelle seguenti cartelle:

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/Arrays*.

Informazioni su aggiunta e rimozione di elementi

Un array contiene elementi e ogni elemento ha una posizione numerica (l'indice) che corrisponde alla modalità con cui si fa riferimento alla posizione di ogni elemento nell'array. Ogni elemento può contenere dati o essere vuoto. I dati contenuti possono essere del formato numerico, stringa, booleano o essere un array o un oggetto.

Quando si creano elementi in un array, si consiglia di creare gli indici in modo sequenziale per facilitare il debug delle applicazioni. In “[Informazioni sul riferimento alla lunghezza e su come determinarla](#)” a pagina 133 si è visto che se si assegna un solo valore in un array all'indice 5, la lunghezza dell'array restituisce 6. Nell'array vengono quindi inseriti cinque valori non definiti.

L'esempio seguente dimostra come creare un nuovo array, eliminare un elemento da un determinato indice e aggiungere e sostituire dati in corrispondenza di un indice di un array:

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
delete monthArr[5];
trace(monthArr); // Jan,Feb,Mar,Apr,May,undefined,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
monthArr[5] = "JUN";
trace(monthArr); // Jan,Feb,Mar,Apr,May,JUN,Jul,Aug,Sep,Oct,Nov,Dec
```

Anche se è stato eliminato l'elemento all'indice 5 dell'array, la lunghezza dell'array rimane 12 e l'elemento all'indice 5 dell'array viene modificato in una stringa vuota anziché essere eliminato totalmente.

È possibile trovare un file di origine di esempio, array.fla, nella cartella Samples sul disco rigido. L'esempio illustra la gestione di array con ActionScript. Il codice dell'esempio crea un array; quindi ordina, aggiunge e rimuove le voci di due componenti List. Il file di esempio si trova nelle seguenti cartelle:

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\Components\ComponentsApplication*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/Arrays*.

Creazione di array indicizzati

Gli array indicizzati memorizzano una serie di uno o più valori. Gli elementi possono essere ricercati in base alla loro posizione nell'array, come probabilmente già fatto nelle sezioni precedenti. Il primo indice è sempre il numero 0 e viene incrementato di un'unità a ogni elemento successivo aggiunto all'array. Per creare un array indicizzato, chiamare la funzione di costruzione della classe Array o inizializzare l'array con un valore letterale array. Entrambe le modalità di creazione sono dimostrate nell'esempio seguente.

Per creare un array indicizzato:

1. Creare un nuovo documento Flash e salvarlo come **indexArray.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var myArray:Array = new Array();
myArray.push("one");
myArray.push("two");
myArray.push("three");
trace(myArray); // one,two,three
```

Nella prima riga del codice ActionScript viene definito un nuovo array per contenere i valori.

3. Selezionare Controllo > Prova filmato per provare il codice.

Il testo seguente viene visualizzato nel pannello Output:

one,two,three

4. Tornare all'ambiente di creazione ed eliminare il codice nel pannello Azioni.

5. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var myArray:Array = ["one", "two", "three"];
trace(myArray); // one,two,three
```

In questo codice si utilizza il valore letterale array per definire un nuovo array per il codice.

Questo codice è equivalente a quello scritto al punto 2. Quando si prova il codice,

l'output visualizzato nel pannello Output è lo stesso.

Creazione di array multidimensionali

In ActionScript è possibile implementare array *nidificati*, ovvero array di array. Questo tipo di array, detto anche *array multidimensionale* può essere paragonato a una matrice o una griglia e può pertanto essere utilizzato nella programmazione per rappresentare questi tipi di strutture. Una scacchiera, ad esempio, è una griglia di otto colonne e righe e può essere rappresentata da un array contenente otto elementi, ognuno dei quali è a sua volta un array che contiene otto elementi.

Prendere in considerazione, ad esempio, un elenco di attività memorizzato come array indicizzato di stringhe:

```
var tasks:Array = ["wash dishes", "take out trash"];
```

Se si desidera memorizzare un elenco separato di attività per ogni giorno della settimana, è possibile creare un array multidimensionale con un elemento per ogni giorno della settimana. Ogni elemento contiene un array indicizzato che a sua volta contiene l'elenco di attività.

ATTENZIONE

Quando si utilizza l'operatore di accesso agli array, il compilatore di ActionScript non è in grado di verificare se l'elemento a cui si accede è una proprietà valida dell'oggetto.

Per creare un array multidimensionale e recuperare elementi dall'array:

1. Creare un nuovo documento Flash e salvarlo come **multiArray1.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var twoDArray:Array = new Array(new Array("one", "two"), new
    Array("three", "four"));
trace(twoDArray);
```

Questo array, `twoDArray`, comprende due elementi che sono a loro volta array che contengono due elementi. In questo caso, `twoDArray` è l'array principale che contiene due array nidificati.

3. Selezionare Controllo > Prova filmato per provare il codice. I dati seguenti saranno visualizzati nel pannello Output:

`one, two, three, four`

4. Tornare allo strumento di creazione e aprire il pannello Azioni. Impostare l'istruzione `trace` come commento, come nell'esempio seguente:

```
// trace(twoDArray);
```

- 5.** Aggiungere il seguente codice ActionScript dopo il codice del fotogramma 1 della linea temporale:

```
trace(twoDArray[0][0]); // uno  
trace(twoDArray[1][1]); // quattro
```

Per recuperare elementi di un array multidimensionale, utilizzare gli operatori di accesso agli array ([]) dopo il nome dell'array di primo livello. Il primo [] fa riferimento all'indice dell'array di primo livello. Gli operatori di accesso agli array successivi fanno riferimento agli elementi degli array nidificati.

- 6.** Selezionare Controllo > Prova filmato per provare il codice. I dati seguenti saranno visualizzati nel pannello Output:

```
one  
four
```

Per creare array multidimensionali è possibile utilizzare cicli for nidificati, come illustrato nell'esempio seguente.

Per creare un array multidimensionale utilizzando un ciclo for:

1. Creare un nuovo documento Flash e salvarlo come **multiArray2.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var gridSize:Number = 3;  
var mainArr:Array = new Array(gridSize);  
var i:Number;  
var j:Number;  
for (i = 0; i < gridSize; i++) {  
    mainArr[i] = new Array(gridSize);  
    for (j = 0; j < gridSize; j++) {  
        mainArr[i][j] = "[" + i + "][" + j + "]";  
    }  
}  
trace(mainArr);
```

Questo codice ActionScript crea un array 3 x 3 e imposta il valore di ogni nodo dell'array al relativo indice. L'array (mainArr) viene quindi tracciato.

- 3.** Selezionare Controllo > Prova filmato per provare il codice.

I dati seguenti saranno visualizzati nel pannello Output:

```
[0][0],[0][1],[0][2],[1][0],[1][1],[1][2],[2][0],[2][1],[2][2]
```

Per eseguire iterazioni su elementi di un array multidimensionale, è anche possibile utilizzare cicli for nidificati, come nell'esempio che segue.

Per utilizzare un ciclo for per eseguire iterazioni su un array multidimensionale:

1. Creare un nuovo documento Flash e salvarlo come **multiArray3.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Dall'esempio precedente
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
    mainArr[i] = new Array(gridSize);
    for (j = 0; j < gridSize; j++) {
        mainArr[i][j] = "[" + i + "][" + j + "]";
    }
}
```

In questo codice, riportato nell'esempio precedente, il ciclo esterno esegue un'iterazione su ogni elemento di `mainArr`. Il ciclo interno esegue un'iterazione su ogni array nidificato e fornisce come output ogni nodo dell'array.

3. Aggiungere il seguente codice ActionScript al fotogramma 1 della linea temporale, di seguito al codice aggiunto al punto 2:

```
// Esegue un'iterazione sugli elementi
var outerArrayLength:Number = mainArr.length;
for (i = 0; i < outerArrayLength; i++) {
    var innerArrayLength:Number = mainArr[i].length;
    for (j = 0; j < innerArrayLength; j++) {
        trace(mainArr[i][j]);
    }
}
```

Questo codice ActionScript esegue iterazioni sugli elementi dell'array. Come condizione del ciclo viene utilizzata la proprietà `length` di ogni array.

4. Selezionare Controllo > Prova filmato per visualizzare gli elementi riportati nel pannello Output. I dati seguenti verranno visualizzati nel pannello Output:

```
[0][0]
[0][1]
[0][2]
[1][0]
[1][1]
[1][2]
[2][0]
[2][1]
[2][2]
```

Per ulteriori informazioni sull'uso degli array, vedere “[Uso degli array](#)” a pagina 130. Per informazioni sugli elementi degli array, vedere “[Informazioni su aggiunta e rimozione di elementi](#)” a pagina 134. Per informazioni sull'operatore di accesso agli array, vedere “[Uso dell'operatore punto e dell'operatore di accesso agli array](#)” a pagina 151.

Creazione di array associativi

Un array associativo, analogo a un oggetto, è composto da *chiavi* e *valori* non ordinati e utilizza le chiavi al posto degli indici numerici per organizzare i valori. Ogni chiave è una stringa univoca e viene utilizzata per accedere al valore a cui è associata. Il valore può essere un tipo di dati Number, Array, Object e così via. Quando si crea codice per rilevare il valore associato a una chiave, si crea un indice o si esegue una *ricerca*. Si tratta dell'uso più comune degli array associativi.

In un'associazione tra una chiave e un valore si dice in genere che la chiave e il valore sono *mappati*. Una rubrica può essere considerata un array associativo dove le chiavi e i valori sono rappresentati, rispettivamente, dai nomi e dagli indirizzi e-mail.



Gli array associativi sono insiemi di coppie chiave/valore non ordinati. Le chiavi di un array associativo non sono mai in un ordine specifico.

Quando si utilizzano array associativi è possibile chiamare l'elemento dell'array desiderato utilizzando una stringa anziché un numero. La stringa è spesso più facile da ricordare. Lo svantaggio è rappresentato dal fatto che questi array non sono utili in un ciclo perché non utilizzano numeri come valori di indici. Sono invece utili quando si deve cercare frequentemente i valori delle chiavi. Se ad esempio si dispone di un array di nomi ed età a cui si fa spesso riferimento, è possibile utilizzare un array associativo.

L'esempio seguente dimostra come creare un oggetto e definire una serie di proprietà in un array associativo.

Per creare un array associativo semplice:

1. Creare un nuovo documento Flash.
2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
// Definisce l'oggetto da utilizzare come array associativo
var someObj:Object = new Object();
// Definisce una serie di proprietà
someObj.myShape = "Rectangle";
someObj.myW = 480;
someObj.myH = 360;
someObj.myX = 100;
someObj.myY = 200;
someObj.myAlpha = 72;
someObj.myColor = 0x00FFFF;
// Visualizza una proprietà utilizzando l'operatore punto e la sintassi
// di accesso agli array
trace(someObj.myAlpha); // 72
trace(someObj["myAlpha"]); // 72
```

La prima riga del codice ActionScript definisce un nuovo oggetto (`someObj`) che viene utilizzato come array associativo. Di seguito viene definita una serie di proprietà in `someObj`. Infine si visualizza una proprietà che si seleziona utilizzando sia l'operatore punto che la sintassi di accesso agli array.

NOTA

È possibile accedere alle variabili in un array associativo utilizzando due metodi diversi:
la sintassi del punto (`someObj.myColor`) e la sintassi degli array (`someObj['myColor']`).

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

Nel pannello Output viene visualizzato due volte il numero 72 che rappresenta i due livelli alfa tracciati.

In ActionScript 2.0 gli array associativi possono essere creati in due modi diversi:

- tramite una funzione di costruzione Object
- tramite una funzione di costruzione Array

Entrambe le modalità di creazione vengono dimostrate negli esempi seguenti.

NOTA

L'esempio precedente utilizza una funzione di costruzione Object per creare un array associativo.

In questo caso è possibile avvalersi di un valore letterale oggetto per inizializzare l'array. Un'istanza della classe Object, detta anche oggetto generico, è identica dal punto di vista funzionale a un array associativo. Le istanze di Object sono infatti praticamente array associativi. Gli array associativi possono essere utilizzati per ottenere funzionalità simili a quelle di un dizionario, nei casi in cui è più comodo basarsi su chiavi in formato stringa anziché su indici numerici. Ogni nome di proprietà dell'oggetto generico funge da chiave che fornisce accesso a un valore memorizzato. Per ulteriori informazioni sui valori letterali, vedere ["Informazioni sui valori letterali"](#) a pagina 94. Per ulteriori informazioni sulle classi, consultare il [Capitolo 6, "Classi"](#) a pagina 197.

Per creare un array associativo tramite una funzione di costruzione di Object:

1. Creare un nuovo documento Flash e salvarlo come **assocArray.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

Questo codice crea un array associativo denominato `monitorInfo` e utilizza un valore letterale oggetto per inizializzare l'array con due coppie chiave/valore.

NOTA

Se non occorre inizializzare l'array quando viene dichiarato, è possibile utilizzare la funzione di costruzione `Object` per creare l'array:

```
var monitorInfo:Object = new Object();
```

3. Selezionare Controllo > Prova filmato.

Il pannello Output visualizza:

Flat Panel, 1600 x 1200

4. Aggiungere il seguente codice ActionScript al fotogramma 1 della linea temporale, di seguito al codice aggiunto in precedenza:

```
monitorInfo["aspectRatio"] = "16:10";
monitorInfo.colors = "16.7 million";
trace(monitorInfo["aspectRatio"] + ", " + monitorInfo.colors);
```

Dopo aver creato l'array con il valore letterale oggetto o con la funzione di costruzione della classe Object, è possibile aggiungervi nuovi valori tramite l'operatore parentesi quadra (`[]`) o l'operatore punto (`.`), come dimostrato nel codice. Il codice appena immesso aggiunge due nuovi valori all'array `monitorInfo`.

5. Selezionare Controllo > Prova filmato.

Il pannello Output visualizza:

16:10, 16.7 million

Tenere presente che una chiave può contenere un carattere spazio. Questa operazione è possibile con l'operatore parentesi quadra, ma genera un errore se si tenta di effettuarla con l'operatore punto. L'uso di spazi nei nomi delle chiavi è sconsigliato. Per ulteriori informazioni sugli operatori parentesi quadra e punto, vedere “[Informazioni sugli operatori](#)” a pagina 143. Per ulteriori informazioni sulla formattazione corretta del codice, vedere “[Formattazione della sintassi ActionScript](#)” a pagina 825.

In alternativa, per creare un array associativo è possibile utilizzare la funzione di costruzione Array e quindi l'operatore parentesi quadra ([]) o l'operatore punto (.) per aggiungere coppie chiave/valore all'array. Se si dichiara un array associativo di tipo Array, non è possibile utilizzare un valore letterale oggetto per inizializzare l'array.

NOTA

L'uso della funzione di costruzione Array per creare un array associativo non presenta vantaggi. Si consiglia di utilizzare questa funzione di costruzione per creare array indicizzati.

L'esempio seguente dimostra come utilizzare la funzione di costruzione Array per creare un array associativo.

Per creare un array associativo tramite la funzione di costruzione di Array:

1. Creare un nuovo documento Flash e salvarlo come **assocArray2.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

Questo codice crea un array associativo denominato `monitorInfo` utilizzando la funzione di costruzione Array e aggiunge una chiave denominata `type` e una denominata `resolution` con i relativi valori.

3. Selezionare Controllo > Prova filmato.

Il pannello Output visualizza:

Flat Panel, 1600 x 1200

NOTA

L'uso della funzione di costruzione Array per creare un array associativo non presenta vantaggi. Si consiglia di utilizzare questa funzione di costruzione per creare array indicizzati.

Gli array associativi sono sostanzialmente istanze della classe Object e non si ottiene alcun vantaggio creando array associativi tramite la funzione di costruzione Array. Anche quando si crea un array associativo utilizzando la funzione di costruzione new Array(), non è possibile utilizzare alcuno dei metodi e delle proprietà della classe Array (come sort() o length) quando si utilizza un array associativo. Se si desidera utilizzare coppie valore/chiave al posto di un indice numerico, è consigliabile utilizzare la classe Object al posto di un array associativo.

Informazioni sugli operatori

In questa sezione sono descritte regole generali relative ai tipi di operatori più comuni, oltre alle regole di precedenza e di associatività di questi ultimi.

Gli operatori sono caratteri che specificano la modalità di combinazione, confronto o modifica dei valori di un'espressione. Per espressione si intende qualsiasi istruzione che Flash è in grado di valutare e che restituisce un valore. È possibile creare un'espressione combinando operatori e valori oppure chiamando una funzione. Per ulteriori informazioni sulle espressioni, vedere “[Informazioni sulla sintassi, le istruzioni e le espressioni](#)” a pagina 76.

Un'espressione matematica, ad esempio, utilizza operatori numerici per manipolare i valori. Alcuni esempi di caratteri operatore sono +, <, * e =. Un'espressione è composta da operatori e *operandi* ovvero da qualsiasi combinazione valida di simboli ActionScript che rappresentano un valore. Un operando è la parte di codice su cui l'operatore esegue le azioni. Ad esempio, nell'espressione x + 2, x e 2 sono operandi e + è un operatore.

Nel codice le espressioni e gli operatori vengono utilizzati di frequente. È possibile creare un'espressione combinando operatori e valori oppure chiamare una funzione.

NOTA

Questa sezione descrive come utilizzare gli operatori, tuttavia non è possibile trattare in modo esaustivo ogni singolo operatore in questa sede. Per informazioni su ogni operatore, inclusi gli operatori speciali che non rientrano nelle categorie trattate, vedere la *Guida di riferimento di ActionScript 2.0*.

Le parti del codice sulle quali l'operatore esegue azioni sono detti *operandi*. È possibile ad esempio utilizzare l'operatore di addizione (+) per sommare i valori di un valore letterale numerico, ad esempio per sommare il valore di una variabile denominata myNum.

myNum + 3;

In questo esempio, myNum e 3 sono operandi.

In questa sezione sono descritte regole generali relative ai tipi di operatori più comuni, oltre alle relative regole di priorità e di associatività:

- “Uso di operatori per la manipolazione di valori” a pagina 145
- “Informazioni sulla priorità e l’associatività degli operatori” a pagina 146
- “Informazioni sull’uso degli operatori con le stringhe” a pagina 149
- “Uso dell’operatore punto e dell’operatore di accesso agli array” a pagina 151
- “Informazioni sugli operatori in forma suffissa” a pagina 153
- “Informazioni sugli operatori unari” a pagina 154
- “Informazioni sugli operatori moltiplicativi” a pagina 155
- “Informazioni sugli operatori additivi” a pagina 155
- “Uso degli operatori numerici” a pagina 155
- “Informazioni sugli operatori relazionali” a pagina 157
- “Informazioni sugli operatori di uguaglianza” a pagina 157
- “Uso degli operatori relazionali e di uguaglianza” a pagina 158
- “Informazioni sugli operatori di assegnazione” a pagina 161
- “Uso degli operatori di assegnazione” a pagina 162
- “Informazioni sugli operatori logici” a pagina 162
- “Uso degli operatori logici” a pagina 163
- “Informazioni sugli operatori di spostamento bit a bit” a pagina 164
- “Informazioni sugli operatori logici bit a bit” a pagina 165
- “Uso degli operatori bit a bit” a pagina 165
- “Informazioni sull’operatore condizionale” a pagina 167
- “Uso degli operatori in un documento” a pagina 167

Per informazioni sugli operatori che non rientrano in queste categorie, vedere la *Guida di riferimento di ActionScript 2.0* che contiene informazioni su tutti gli operatori disponibili.

Le sezioni seguenti mostrano alcuni usi comuni degli operatori. Per ulteriori informazioni sull’uso di molti operatori in un solo esempio di codice, vedere “[Uso degli operatori in un documento](#)” a pagina 167.

Uso di operatori per la manipolazione di valori

Gli operatori vengono normalmente utilizzati per manipolare valori in Flash, ad esempio per creare un gioco dove il punteggio cambia in seguito all'interazione dell'utente con istanze sullo stage. È possibile utilizzare una variabile per memorizzare un valore e operatori per manipolare il valore della variabile.

Potrebbe essere necessario, ad esempio, incrementare il valore di una variabile denominata `myScore`. L'esempio seguente dimostra l'uso degli operatori `+` (addizione) e `+=` (assegnazione addizione) per aggiungere e incrementare valori nel codice.

Per manipolare valori tramite gli operatori:

1. Creare un nuovo documento Flash.
2. Aprire il pannello Azioni (Finestra > Azioni) e immettere il seguente codice nel pannello Script:

```
// Esempio uno
var myScore:Number = 0;
myScore = myScore + 1;
trace("Example one: " + myScore); // 1

// Esempio due
var secondScore:Number = 1;
secondScore += 3;
trace("Example two: " + secondScore); // 4
```

3. Selezionare Controllo > Prova filmato.

Il pannello Output visualizza:

```
Example one: 1
Example two: 4
```

L'uso dell'operatore addizione è piuttosto semplice in quanto consente di sommare due valori. Nel primo esempio di codice, il valore corrente di `myScore` viene sommato al numero 1 e il risultato viene memorizzato nella variabile `myScore`.

Il secondo esempio di codice utilizza l'operatore di assegnazione di addizione per sommare e assegnare un nuovo valore in un solo passaggio. La riga `myScore = myScore + 1` dell'esercizio precedente può essere riscritta come `myScore++` oppure `myScore += 1`.

L'operatore di incremento (`++`) crea lo stesso risultato di `myScore = myScore + 1`, ma in modo più semplice, perché gestisce un incremento e un'assegnazione simultaneamente. Nel codice ActionScript seguente è presente un esempio dell'operatore di incremento:

```
var myNum:Number = 0;
myNum++;
trace(myNum); // 1
myNum++;
trace(myNum); // 2
```

Nel frammento di codice precedente non sono presenti operatori di assegnazione, ma solo un operatore di incremento.

Il valore di una variabile può essere manipolato tramite operatori per tutto il tempo in cui una condizione è true. È possibile, ad esempio, ricorrere all'operatore di incremento (++) per incrementare la variabile `i` mentre la condizione è soddisfatta. Nel codice seguente, la condizione è true quando `i` è inferiore a 10. In questo caso è possibile incrementare `i` di un numero utilizzando `i++`.

```
var i:Number;  
for (i = 1; i < 10; i++) {  
    trace(i);  
}
```

Il pannello Output visualizza i numeri da 1 a 9, vale a dire l'incremento del valore di `i` fino al raggiungimento della condizione di fine (`i` uguale a 10). L'ultimo valore visualizzato è 9. Il valore di `i` è pertanto 1 all'inizio della riproduzione del file SWF e 9 al completamento della traccia.

Per ulteriori informazioni su condizioni e cicli, vedere “[Informazioni sulle istruzioni](#)” [a pagina 106](#).

Informazioni sulla priorità e l'associatività degli operatori

Quando nella stessa istruzione vengono utilizzati due o più operatori, alcuni di essi hanno la priorità su altri. La priorità e l'associatività degli operatori determina l'ordine in cui questi vengono elaborati. ActionScript prevede una gerarchia in base alla quale si determina quali operatori vengono eseguiti prima di altri. Per informazioni, vedere la tabella alla fine della sezione.

Se si ha familiarità con le operazioni matematiche o la programmazione di base può sembrare ovvio che il compilatore elabori l'operatore di moltiplicazione (*) prima di quello di addizione (+). Il compilatore necessita tuttavia di istruzioni esplicite sull'ordine di elaborazione degli operatori. Queste istruzioni vengono dette globalmente *priorità degli operatori*.

Un esempio è dato dall'uso congiunto degli operatori di moltiplicazione e addizione:

```
var mySum:Number;  
mySum = 2 + 4 * 3;  
trace(mySum); // 14
```

Il risultato dell'istruzione è 14, perché la moltiplicazione ha una priorità superiore. $4 * 3$ viene pertanto elaborato prima e il risultato viene sommato a 2.

Per controllare l'elaborazione è possibile racchiudere tra parentesi le espressioni. ActionScript definisce un ordine di priorità predefinito per gli operatori che è possibile modificare tramite l'operatore parentesi tonda (()). Se l'operazione di addizione viene inserita tra parentesi, viene eseguita prima della moltiplicazione:

```
var mySum:Number;  
mySum = (2 + 4) * 3;  
trace(mySum); // 18
```

Il risultato dell'istruzione ora è 18.

Gli operatori possono anche avere la stessa priorità. In questo caso l'ordine di elaborazione è dato dall'associatività. L'associatività può operare da sinistra a destra o da destra a sinistra.

Considerare nuovamente l'operatore moltiplicazione. L'associatività di questo operatore è da sinistra a destra, pertanto le due istruzioni seguenti sono analoghe.

```
var mySum:Number;  
var myOtherSum:Number;  
mySum = 2 * 4 * 3;  
myOtherSum = (2 * 4) * 3;  
trace(mySum); // 24  
trace(myOtherSum); // 24
```

In alcune situazioni, nella stessa espressione possono essere presenti due o più operatori con la stessa priorità. In questi casi, il compilatore utilizza le regole di *associatività* per determinare l'operatore da elaborare per primo. Tutti gli operatori binari, ad eccezione degli operatori di assegnazione, hanno un'associatività *da sinistra a destra*, ovvero gli operatori a sinistra vengono elaborati prima di quelli a destra. Gli operatori binari di assegnazione e l'operatore condizionale (?:) hanno un'associatività *da destra a sinistra*, ovvero gli operatori a destra vengono elaborati prima di quelli a sinistra. Per ulteriori informazioni sugli operatori di assegnazione, vedere “[Uso degli operatori di assegnazione](#)” a pagina 162. Per ulteriori informazioni sull'operatore condizionale (?:), vedere “[Informazioni sull'operatore condizionale](#)” a pagina 167.

Considerare, ad esempio, gli operatori minore di (<) e maggiore di (>) che hanno la stessa priorità. Se entrambi gli operatori vengono utilizzati nella stessa espressione, l'operatore a sinistra viene elaborato per primo, perché l'associatività di entrambi è da sinistra a destra. Le due istruzioni seguenti producono pertanto lo stesso risultato:

```
trace(3 > 2 < 1); // false  
trace((3 > 2) < 1); // false
```

L'operatore maggiore di (>) viene elaborato per primo e restituisce `true`, perché l'operando 3 è maggiore dell'operando 2. Il valore `true` viene quindi passato all'operatore minore di (<) insieme all'operando 1. L'operatore minore di (<) converte il valore `true` al valore numerico 1 e confronta questo valore numerico con il secondo operando 1, restituendo il valore `false` perché il valore di 1 non è minore di 1.

Prestare attenzione all'ordine degli operandi nel codice ActionScript, in particolare quando si creano condizioni complesse e si conosce la frequenza del valore true di una delle condizioni. Se, ad esempio, si sa che nella condizione `i` sarà maggiore di 50, scrivere in primo luogo `i < 50`. In questo modo la condizione viene verificata per prima e la seconda condizione non deve essere verificata molto spesso.

La tabella seguente contiene l'elenco di tutti gli operatori di ActionScript in ordine di priorità, dalla maggiore alla minore, e ne specifica l'associatività. Per ulteriori informazioni e linee guida sull'utilizzo degli operatori e delle parentesi, consultare il [Capitolo 19, “Formattazione della sintassi ActionScript” a pagina 825](#).

Operatore	Descrizione	Associatività
Priorità più alta		
<code>x++</code>	Incremento dopo l'operazione	Da sinistra a destra
<code>x--</code>	Decremento dopo l'operazione	Da sinistra a destra
<code>.</code>	Accesso alle proprietà di un oggetto	Da sinistra a destra
<code>[]</code>	Elemento di un array	Da sinistra a destra
<code>()</code>	Parentesi	Da sinistra a destra
<code>function ()</code>	Chiamata di una funzione	Da sinistra a destra
<code>++x</code>	Incremento prima dell'operazione	Da destra a sinistra
<code>--x</code>	Decremento prima dell'operazione	Da destra a sinistra
<code>-</code>	Negazione unaria, ad esempio <code>x = -1</code>	Da sinistra a destra
<code>~</code>	NOT bit a bit	Da destra a sinistra
<code>!</code>	NOT logico	Da destra a sinistra
<code>new</code>	Allocazione di un oggetto	Da destra a sinistra
<code>delete</code>	Disallocazione di un oggetto	Da destra a sinistra
<code>typeof</code>	Tipo di oggetto	Da destra a sinistra
<code>void</code>	Restituzione di un valore non definito	Da destra a sinistra
<code>*</code>	Moltiplicazione	Da sinistra a destra
<code>/</code>	Divisione	Da sinistra a destra
<code>%</code>	Modulo	Da sinistra a destra
<code>+</code>	Più unario	Da destra a sinistra
<code>-</code>	Meno unario	Da destra a sinistra
<code><<</code>	Spostamento a sinistra bit a bit	Da sinistra a destra

Operatore	Descrizione	Associatività
>>	Spostamento a destra bit a bit	Da sinistra a destra
>>>	Spostamento a destra bit a bit (senza segno)	Da sinistra a destra
instanceof	Istanza di (trova la classe di cui l'oggetto è un'istanza) Richiede Flash Player 6 o superiore	Da sinistra a destra
<	Minore di	Da sinistra a destra
<=	Minore o uguale a	Da sinistra a destra
>	Maggiore di	Da sinistra a destra
>=	Maggiore o uguale a	Da sinistra a destra
==	Uguale	Da sinistra a destra
!=	Non uguale	Da sinistra a destra
&	AND bit a bit	Da sinistra a destra
^	XOR bit a bit	Da sinistra a destra
	OR bit a bit	Da sinistra a destra
&&	AND logico	Da sinistra a destra
	OR logico	Da sinistra a destra
? :	Condizionale	Da destra a sinistra
=	Assegnazione	Da destra a sinistra
*=, /=, %=, +=, -=, &=, =, ^=, <<=, >>=, >>>=	Assegnazioni composte	Da destra a sinistra
,	Virgola	Da sinistra a destra
Priorità più bassa		

Informazioni sull'uso degli operatori con le stringhe

Gli operatori di confronto considerano le stringhe soltanto se entrambi gli operandi sono stringhe. Un'eccezione alla regola è rappresentata dall'operatore di uguaglianza rigorosa (==). Se un solo operando è una stringa, ActionScript converte entrambi gli operandi in numeri ed esegue un confronto numerico. Per ulteriori informazioni sugli operatori numerici, vedere ["Uso degli operatori numerici" a pagina 155.](#)

Ad eccezione dell'operatore di uguaglianza (==), gli operatori di confronto (>, >=, < e <=) lavorano in modo diverso con le stringhe rispetto a quanto avviene con altri tipi di valori.

Gli operatori di confronto esaminano le stringhe per determinare qual è la prima in ordine alfabetico. Le stringhe a caratteri maiuscoli precedono quelle a caratteri minuscoli. "Egg", ad esempio, precede "chicken".

```
var c:String = "chicken";
var e:String = "Egg";
trace(c < e); // false
var riddleArr:Array = new Array(c, e);
trace(riddleArr); // chicken,Egg
trace(riddleArr.sort()); // Egg,chicken
```

Nel codice ActionScript riportato, il metodo `sort()` della classe `Array` dispone il contenuto dell'array in ordine alfabetico. Il valore "Egg" viene riportato prima del valore "chicken" perché la E maiuscola ha la precedenza rispetto alla c minuscola. Per confrontare le stringhe indipendentemente dalle maiuscole e dalle minuscole, convertirle tutte a lettere maiuscole o minuscole prima di eseguire il confronto. Per ulteriori informazioni sugli operatori di confronto, vedere ["Informazioni sugli operatori di uguaglianza" a pagina 157](#) e ["Uso degli operatori relazionali e di uguaglianza" a pagina 158](#).

Per convertire tutte le stringhe a lettere minuscole o maiuscole prima del confronto, è possibile ricorrere ai metodi `toLowerCase()` o `toUpperCase()`. Nell'esempio seguente, entrambe le stringhe vengono convertite a caratteri minuscoli e confrontate; chicken viene ora riportato prima di egg:

```
var c:String = "chicken";
var e:String = "Egg";
trace(c.toLowerCase() < e.toLowerCase()); // true
```

NOTA

Gli operatori di confronto consentono di confrontare solo due stringhe. Il confronto non può essere eseguito se, ad esempio, un operando ha un valore numerico. Se un operando è una stringa, ActionScript converte entrambi gli operandi in numeri ed esegue un confronto numerico.

È possibile utilizzare gli operatori per manipolare le stringhe, ad esempio l'operatore di addizione (+) per concatenare operandi in formato stringa, come probabilmente è già stato fatto nelle istruzioni `trace`. Si osservi ad esempio il seguente codice:

```
var myNum:Number = 10;
trace("The variable is " + myNum + ".");
```

Quando si prova il codice, nel pannello Output viene visualizzato quanto segue:

The variable is 10.

Nell'esempio precedente, l'istruzione `trace` utilizza l'operatore + per concatenare anziché sommare. Quando si lavora con stringhe e numeri, Flash a volta concatena i valori anziché sommarli.

È possibile ad esempio concatenare due stringhe di variabili diverse in un unico campo di testo. Nel codice ActionScript seguente, la variabile myNum viene concatenata con una stringa e il risultato viene visualizzato nel campo di testo myTxt sullo stage.

```
this.createTextField("myTxt", 11, 0, 0, 100, 20);
myTxt.autoSize = "left";
var myNum:Number = 10;
myTxt.text = "One carrot. " + myNum + " large eggplants.";
myTxt.text += " Lots of vegetable broth.;"
```

Questo codice restituisce quanto seguente in un campo di testo con il nome di istanza myTxt:
One carrot. 10 large eggplants. Lots of vegetable broth.

L'esempio precedente mostra come utilizzare gli operatori di addizione (+) e di assegnazione di addizione (+=) per concatenare stringhe. Nella terza riga del codice l'operatore di addizione viene utilizzato per concatenare il valore della variabile myNum nel campo di testo, mentre nella quarta riga di codice l'operatore di assegnazione di addizione viene utilizzato per concatenare una stringa al valore esistente del campo di testo.

Se solo uno degli operandi della stringa di testo è una stringa, Flash converte in stringa anche l'altro operando. Il valore di myNum viene pertanto convertito in una stringa nell'esempio precedente.



In ActionScript gli spazi all'inizio o alla fine della stringa sono considerati parte integrante della stringa.

Uso dell'operatore punto e dell'operatore di accesso agli array

È possibile utilizzare l'operatore punto (.) e l'operatore di accesso agli array ([]) per accedere alle proprietà ActionScript incorporate o personalizzate. Gli operatori punto consentono di fare riferimento a determinati indici di un oggetto. Se, ad esempio, un oggetto contiene alcune informazioni relative all'utente, è possibile specificare un nome di chiave nell'operatore di accesso agli array per recuperare il nome di un utente, come dimostrato nel seguente codice ActionScript:

```
var someUser:Object = {name:"Hal", id:2001};
trace("User's name is: " + someUser["name"]); // Nome utente: Hal
trace("User's id is: " + someUser["id"]); // ID utente: 2001
```

Il codice ActionScript seguente, ad esempio, utilizza l'operatore punto per impostare alcune proprietà all'interno degli oggetti:

```
myTextField.border = true;
year.month.day = 9;
myTextField.text = "My text";
```

L'operatore punto e l'operatore di accesso agli array sono molto simili. L'operatore punto accetta come proprietà un identificatore, mentre l'operatore di accesso agli array restituisce il contenuto come nome e quindi accede al valore di tale proprietà denominata. L'operatore di accesso agli array consente di impostare e recuperare in modo dinamico nomi di istanza e variabili.

L'operatore di accesso agli array è utile se non si conoscono le chiavi presenti in un oggetto. In questo caso è possibile utilizzare un ciclo `for .. in` per eseguire iterazioni su un oggetto o clip filmato e visualizzarne il contenuto.

Per utilizzare l'operatore punto e l'operatore di accesso agli array:

1. In un nuovo documento Flash, creare un clip filmato sulla linea temporale principale.
2. Selezionare il clip filmato e aprire la finestra di ispezione Proprietà.
3. Specificare il nome di istanza `myClip`.
4. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
myClip.spam = 5;  
trace(myClip.spam); // 5
```

Per impostare un valore nell'istanza `myClip` sulla linea temporale corrente, è possibile utilizzare l'operatore punto o l'operatore di accesso agli array, come dimostrato di seguito. Se si scrive un'espressione all'interno dell'operatore di accesso agli array, questa espressione viene valutata e il risultato viene utilizzato come nome della variabile.

5. Selezionare Controllo > Prova filmato per provare il documento.

Nel pannello Output viene visualizzato 5.

6. Tornare all'ambiente di creazione e sostituire la prima riga del codice ActionScript con la seguente:

```
myClip["spam"] = 10;
```

7. Selezionare Controllo > Prova filmato per provare il documento.

Nel pannello Output viene visualizzato 10.

8. Tornare all'ambiente di creazione e fare doppio clic sull'istanza `myClip`.

9. Aggiungere quattro nuove istanze all'interno di `myClip`.

10. Tramite la finestra di ispezione Proprietà aggiungere i seguenti nomi di istanza per le quattro istanze: `nestedClip1`, `nestedClip2`, `nestedClip3` e `nestedClip4`.

11. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
var i:Number;  
for (i = 1; i <= 4; i++) {  
    myClip["nestedClip" + i]._visible = false;  
}
```

Questo codice ActionScript consente di mostrare e nascondere ognuno dei clip filmato nidificati.

12. Selezionare Controllo > Prova filmato per provare il codice ActionScript appena aggiunto.

Le quattro istanze nidificate sono ora invisibili. L'operatore di accesso agli array viene utilizzato per eseguire iterazioni su ogni clip filmato nidificato nell'istanza myClip e impostare la relativa proprietà visible in modo dinamico. In questo modo si risparmia tempo, perché non occorre fare riferimento in modo specifico a ogni istanza.

È inoltre possibile utilizzare l'operatore di accesso agli array a sinistra di un'assegnazione per impostare il nome dell'istanza, della variabile e dell'oggetto in modo dinamico:

```
myNum[i] = 10;
```

In ActionScript 2.0 l'operatore parentesi quadra consente di accedere alle proprietà di un oggetto create in modo dinamico se la definizione di classe dell'oggetto non prevede l'attributo `dynamic`, e di creare array multidimensionali. Per ulteriori informazioni sulla creazione di array multidimensionali tramite gli operatori di accesso agli array, vedere “[Creazione di array multidimensionali](#)” a pagina 136.

Informazioni sugli operatori in forma suffissa

Gli operatori in forma suffissa accettano un operatore e ne incrementano o decrementano il valore. Sebbene si tratti di operatori unari, sono classificati separatamente dal resto degli operatori di questo tipo perché hanno una priorità maggiore e si comportano in modo particolare. Per ulteriori informazioni sugli operatori unari, vedere “[Informazioni sugli operatori unari](#)” a pagina 154.

Quando si utilizza un operatore in forma suffissa all'interno di un'espressione complessa, il valore dell'espressione viene restituito prima dell'elaborazione dell'operatore in forma suffissa. Nel codice seguente, ad esempio, il valore dell'espressione `xNum++` viene restituito prima che il valore venga incrementato.

```
var xNum:Number = 0;  
trace(xNum++); // 0  
trace(xNum); // 1
```

Quando si traccia il codice, nel pannello Output viene visualizzato il seguente testo:

0
1

Gli operatori in questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
<code>++</code>	Incremento (in forma suffissa)
<code>--</code>	Decremento (in forma suffissa)

Informazioni sugli operatori unari

Gli operatori unari accettano un operando. Gli operatori di incremento (`++`) e decremento (`-`) di questo gruppo sono operatori *in forma prefissa*, ovvero vengono utilizzati prima dell'operando in un'espressione. Possono anche essere inseriti dopo l'operando e in tal caso diventano operatori *in forma suffissa*. Per ulteriori informazioni sugli operatori in forma suffissa, vedere “[Informazioni sugli operatori in forma suffissa](#)” a pagina 153.

Gli operatori in forma prefissa sono diversi da quelli in forma suffissa perché l'operazione di incremento o decremento viene completata prima della restituzione del valore di tutta l'espressione. Nel codice seguente, ad esempio, il valore dell'espressione `xNum++` viene restituito dopo che il valore viene incrementato.

```
var xNum:Number = 0;  
trace(++xNum); // 1  
trace(xNum); // 1
```

Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
<code>++</code>	Incremento (in forma prefissa)
<code>--</code>	Incremento (in forma prefissa)
<code>+</code>	+ unario
<code>!</code>	- unario (negazione)
<code>typeof</code>	Restituisce informazioni sul tipo
<code>void</code>	Restituzione di un valore non definito

Informazioni sugli operatori moltiplicativi

Gli operatori moltiplicativi accettano due operandi ed eseguono moltiplicazioni, divisioni e moduli. Gli operatori numerici comprendono gli operatori additivi. Per informazioni sugli operatori additivi, vedere “[Informazioni sugli operatori additivi](#)” a pagina 155.

Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
*	Moltiplicazione
/	Divisione
%	Modulo

Per informazioni sull'uso degli operatori moltiplicativi, vedere “[Uso degli operatori numerici](#)” a pagina 155.

Informazioni sugli operatori additivi

Gli operatori additivi accettano due operandi ed eseguono addizioni o sottrazioni. Gli operatori numerici comprendono gli operatori moltiplicativi. Per informazioni sugli operatori moltiplicativi, vedere “[Informazioni sugli operatori moltiplicativi](#)” a pagina 155.

Gli operatori in questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
+	Addizione
-	Sottrazione

Per informazioni sull'uso degli operatori additivi, vedere “[Uso degli operatori numerici](#)” a pagina 155.

Uso degli operatori numerici

Gli operatori numerici consentono di sommare, sottrarre, dividere e moltiplicare valori in ActionScript. È possibile eseguire diversi tipi di operazioni matematiche. Uno degli operatori più comuni è l'operatore di incremento che in genere ha il formato `i++`. consente di eseguire diverse operazioni. Per ulteriori informazioni sull'operatore di incremento, vedere “[Uso di operatori per la manipolazione di valori](#)” a pagina 145.

È possibile aggiungere l'incremento prima (*preincremento*) o dopo (*postincremento*) un operando.

Per comprendere l'uso degli operatori numerici in ActionScript:

1. Creare un nuovo documento Flash.
2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
// Esempio uno
var firstScore:Number = 29;
if (++firstScore >= 30) {
    // Viene eseguita la traccia
    trace("Success! ++firstScore is >= 30");
}
// Esempio due
var secondScore:Number = 29;
if (secondScore++ >= 30) {
    // Non viene eseguita la traccia
    trace("Success! secondScore++ is >= 30");
}
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

Per il blocco di codice indicato con "Esempio uno" viene eseguita la traccia, mentre per "Esempio due" non viene eseguita. Il primo esempio utilizza un preincremento (`++firstScore`) per incrementare e calcolare `firstScore` prima di eseguire la verifica a fronte di 30. Per questo motivo `firstScore` viene incrementato a 30 e quindi verificato a fronte di 30.

Il secondo esempio invece utilizza un postincremento (`secondScore++`) che viene valutato dopo la verifica. Per questo motivo 29 viene confrontato con 30 e incrementato a 30 dopo la valutazione.

Per ulteriori informazioni sulla priorità degli operatori, vedere ["Informazioni sulla priorità e l'associatività degli operatori"](#) a pagina 146.

Quando si caricano dati da origine esterne (ad esempio file XML, FlashVars, servizi Web e così via), prestare molta attenzione se si lavora con operatori numerici. A volte Flash considera i numeri come stringhe perché il file SWF non è in grado di riconoscere il tipo di dati del numero. In tal caso, la somma di 3 e 7 sarebbe 37 perché entrambi i numeri verrebbero concatenati come stringhe anziché sommati. In questa situazione è necessario convertire manualmente i dati da stringhe in numeri tramite la funzione `Number()`.

Informazioni sugli operatori relazionali

Gli operatori relazionali accettano due operandi, ne confrontano il valore e restituiscono un valore booleano. Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
<	Minore di
>	Maggiore di
<=	Minore o uguale a
>=	Maggiore o uguale a
instanceof	Controlla la catena di prototipi
in	Controlla le proprietà dell'oggetto

Per informazioni sull'uso degli operatori relazionali, vedere “[Uso degli operatori relazionali e di uguaglianza](#)” a pagina 158.

Informazioni sugli operatori di uguaglianza

Gli operatori di uguaglianza accettano due operandi, ne confrontano il valore e restituiscono un valore booleano. Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
==	Uguaglianza
!=	Disuguaglianza
====	Uguaglianza rigorosa
!==	Disuguaglianza rigorosa

Per informazioni sull'uso degli operatori di uguaglianza, vedere “[Uso degli operatori relazionali e di uguaglianza](#)” a pagina 158.

Uso degli operatori relazionali e di uguaglianza

Gli operatori relazionali e di uguaglianza, detti anche *operatori di confronto*, confrontano i valori delle espressioni e restituiscono `true` o `false`, ovvero un valore booleano. Questi operatori vengono spesso utilizzati nelle istruzioni condizionali e nei cicli per specificare la condizione di arresto del ciclo.

È possibile utilizzare l'operatore di uguaglianza (`==`) per determinare se i valori o i riferimenti di due operandi sono uguali. Il confronto restituisce un valore booleano. I valori degli operandi stringa, numerici o booleani vengono confrontati utilizzando un valore. Gli operanti oggetto e array vengono confrontati in base a un riferimento.

Questo esempio dimostra come utilizzare l'operatore di uguaglianza per provare la lunghezza dell'array. Viene visualizzato un messaggio nel pannello Output se l'array non contiene elementi.

```
var myArr:Array = new Array();
if (myArr.length == 0) {
    trace("the array is empty.");
}
```

Selezionando Controllo > Prova filmato, nel pannello Output viene visualizzata la stringa `the array is empty.`

L'operatore di uguaglianza può essere utilizzato per confrontare valori, ma non per impostarli. A volte si tenta di utilizzare l'operatore di assegnazione (`=`) per verificare l'uguaglianza.

Per utilizzare gli operatori relazionali e di uguaglianza nel codice:

1. Creare un nuovo documento Flash.
2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
var myNum:Number = 2;
if (myNum == 2) {
    //fa qualcosa
    trace("It equals 2");
}
```

In questo codice ActionScript viene utilizzato l'operatore di uguaglianza (`==`) per verificare che la variabile `myNum` sia uguale a due.

3. Selezionare Controllo > Prova filmato.

Nel pannello Output viene visualizzata la stringa `It equals 2.`

4. Tornare all'ambiente di creazione e modificare:

```
var myNum:Number = 2;
in:
var myNum:Number = 4;
```

5. Selezionare di nuovo Controllo > Prova filmato.

Nel pannello Output non viene visualizzata la stringa `It equals 2.`

6. Tornare all'ambiente di creazione e modificare:

```
if (myNum == 2) {  
    in  
    if (myNum = 2) {
```

7. Selezionare di nuovo Controllo > Prova filmato.

Nel pannello Output viene di nuovo visualizzata la stringa `It equals 2.`

Al punto 6, anziché confrontare `myNum` a 2, a `myNum` viene assegnato il valore 2. In questo caso, l'istruzione `if` viene eseguita indipendentemente dal valore precedente di `myNum` e si possono verificare risultati imprevisti durante la prova del documento Flash.

Per ulteriori informazioni sull'uso corretto dell'operatore di assegnazione, vedere “[Uso degli operatori di assegnazione](#)” a pagina 162.

L'operatore di uguaglianza rigorosa (`==`) è simile all'operatore di uguaglianza ad eccezione del fatto che non esegue la conversione del tipo. Se i due operandi sono di tipo diverso, l'operatore di uguaglianza restituisce `false`. L'operatore di diseguaglianza rigorosa (`!=`) restituisce il risultato opposto a quello dell'operatore di uguaglianza rigorosa.

Il codice ActionScript seguente dimostra la differenza principale tra l'operatore di uguaglianza (`==`) e l'operatore di uguaglianza rigorosa (`==`):

```
var num1:Number = 32;  
var num2:String = new String("32");  
trace(num1 == num2); // true  
trace(num1 === num2); // false
```

In primo luogo vengono definite le variabili numeriche `num1` e `num2`. Se le variabili vengono confrontate tramite l'operatore di uguaglianza, Flash tenta di convertire i valori nello stesso tipo di dati e quindi confronta i valori per verificarne l'uguaglianza. Se invece si utilizza l'operatore di uguaglianza rigorosa (`==`), Flash non tenta di eseguire alcuna conversione dei tipi di dati prima di confrontare i valori e di conseguenza le variabili vengono considerate due valori distinti.

Nell'esempio seguente, viene utilizzato l'operatore maggiore o uguale a (`>=`) per confrontare i valori ed eseguire il codice in base al valore immesso dall'utente in un campo di testo.

Per utilizzare l'operatore maggiore o uguale a nel codice:

1. Selezionare File > Nuovo, quindi Documento Flash per creare un nuovo file FLA.

2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function(evt_obj:Object):Void {
    var myNum:Number = Number(myTxt.text);
    if (isNaN(myNum)) {
        trace("Please enter a number");
        return;
    }
    if (myNum >= 10) {
        trace("Your number is greater than or equal to 10");
    } else {
        trace("Your number is less than 10");
    }
};
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

È inoltre possibile verificare se alcune condizioni sono soddisfatte ed eseguire un blocco alternativo se la condizione non è soddisfatta.

4. Modificare la condizione nel codice ActionScript come indicato di seguito:

```
if (myNum == 10) {
    trace("Your number is 10");
} else {
    trace("Your number is not 10");
}
```

5. Selezionare Controllo > Prova filmato per provare nuovamente il codice ActionScript.

Fatta eccezione per l'operatore di uguaglianza rigorosa (==), gli operatori di confronto considerano le stringhe solo se entrambi gli operandi sono stringhe. Se solo uno degli operandi è una stringa, entrambi gli operandi vengono convertiti in numeri e viene eseguito un confronto numerico. Per ulteriori informazioni su operatori e stringhe, vedere “[Informazioni sull'uso degli operatori con le stringhe](#)” a pagina 149. Per informazioni sull'effetto dell'ordine e della priorità degli operatori sul codice ActionScript, vedere “[Informazioni sulla priorità e l'associatività degli operatori](#)” a pagina 146.

Informazioni sugli operatori di assegnazione

Gli operatori di assegnazione accettano due operandi e assegnano un valore a uno di essi in base al valore dell'altro operando. Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
=	Assegnazione
*=	Assegnazione moltiplicazione
/=	Assegnazione divisione
%=	Assegnazione modulo
+=	Assegnazione addizione
-=	Assegnazione sottrazione
<<=	Assegnazione spostamento a sinistra bit a bit
>>=	Assegnazione spostamento a destra bit a bit
>>>=	Assegnazione spostamento a destra senza segno bit a bit
&=	Assegnazione AND bit a bit
^=	Assegnazione XOR bit a bit
=	Assegnazione OR bit a bit

Per informazioni sull'uso degli operatori di assegnazione, vedere “[Uso degli operatori di assegnazione](#)” a pagina 162.

Uso degli operatori di assegnazione

L'operatore di assegnazione (=) consente di assegnare un dato valore a una variabile. È possibile assegnare una variabile a una stringa, come segue:

```
var myText:String = "ScratchyCat";
```

È inoltre possibile utilizzare questo operatore per assegnare diverse variabili nella stessa espressione. Nell'istruzione seguente, il valore 10 viene assegnato alle variabili numOne, numTwo e numThree.

```
var numOne:Number;  
var numTwo:Number;  
var numThree:Number;  
numOne = numTwo = numThree = 10;
```

Gli operatori di assegnazione composti consentono, inoltre, di combinare più operazioni. Questi operatori eseguono l'operazione su entrambi gli operandi e assegnano il nuovo valore al primo operando. Ad esempio, le due istruzioni seguenti eseguono la stessa operazione:

```
var myNum:Number = 0;  
myNum += 15;  
myNum = myNum + 15;
```

Quando si utilizza l'operatore di assegnazione, possono presentarsi problemi se si tenta di sommare i valori di un'espressione, come nell'esempio seguente:

```
trace("the sum of 5 + 2 is: " + 5 + 2); // La somma di 5 + 2 è 52
```

Flash concatena i valori 5 e 2 anziché sommarli. Per aggirare il problema, è possibile inserire l'espressione 5+2 all'interno di parentesi, come nel codice seguente:

```
trace("the sum of 5 + 2 is: " + (5 + 2)); // La somma di 5 + 2 è 7
```

Informazioni sugli operatori logici

Gli operatori logici confrontano valori booleani (true e false) e restituiscono un valore booleano in base al confronto. Se, ad esempio, entrambi gli operandi restituiscono true, l'operatore logico AND (&&) restituisce true. Se invece uno o entrambi gli operandi restituiscono true, l'operatore logico OR (||) restituisce true.

Gli operatori logici accettano due operandi e restituiscono un valore booleano. La priorità degli operatori è diversa. Gli operatori sono elencati nella tabella in ordine di priorità, dalla maggiore alla minore.

Operatore	Operazione eseguita
-----------	---------------------

&&	AND logico
----	------------

	OR logico
--	-----------

Per informazioni sull'uso degli operatori logici, vedere [“Uso degli operatori logici” a pagina 163](#).

Uso degli operatori logici

Gli operatori logici vengono spesso utilizzati con gli operatori di confronto per determinare la condizione di un'istruzione if, come dimostrato di seguito.

Per utilizzare gli operatori logici nel codice:

1. Selezionare File > Nuovo e creare un nuovo documento Flash.
2. Aprire il pannello Azioni e digitare il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function():Void {
    var myNum:Number = Number(myTxt.text);
    if (isNaN(myNum)) {
        trace("Please enter a number");
        return;
    }
    if ((myNum > 10) && (myNum < 20)) {
        trace("Your number is between 10 and 20");
    } else {
        trace("Your number is NOT between 10 and 20");
    }
};
```

Questo codice ActionScript crea un campo di testo in fase di runtime. Se si digita un numero nel campo di testo e si fa clic sul pulsante sullo stage, Flash utilizza l'operatore logico per visualizzare un messaggio nel pannello Output. Il messaggio dipende dal valore del numero digitato nel campo di testo.

Prestare attenzione all'ordine degli operandi, in particolare se si scrivono condizioni complesse. Nel frammento di codice seguente, l'operatore AND logico viene utilizzato per controllare che un numero abbia un valore compreso tra 10 e 20. In base al risultato viene visualizzato un messaggio appropriato. Se il numero è minore di 10 o maggiore di 20, viene visualizzato un messaggio alternativo nel pannello Output.

```
submit_mc.onRelease = function():Void {
    var myNum:Number = Number(myTxt.text);
    if (isNaN(myNum)) {
        trace("Please enter a number");
        return;
    }
    if ((myNum > 10) && (myNum < 20)) {
        trace("Your number is between 10 and 20");
    } else {
        trace("Your number is NOT between 10 and 20");
    }
};
```

Informazioni sugli operatori di spostamento bit a bit

Gli operatori di spostamento bit a bit accettano due operandi e spostano i bit del primo operando in base a quanto specificato dal secondo operando. Tutti gli operatori di questa tabella hanno priorità equivalente:

Operatore	Operazione eseguita
<<	Spostamento a sinistra bit a bit
>>	Spostamento a destra bit a bit
>>>	Spostamento a destra bit a bit senza segno

<<	Spostamento a sinistra bit a bit
>>	Spostamento a destra bit a bit
>>>	Spostamento a destra bit a bit senza segno

Per informazioni sull'uso degli operatori bit a bit, vedere “[Uso degli operatori bit a bit](#)” [a pagina 165](#). Per informazioni specifiche su ogni operatore bit a bit, vedere la relativa voce nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sugli operatori logici bit a bit

Gli operatori logici bit a bit accettano due operandi ed eseguono operazioni logiche a livello di bit. La priorità degli operatori è diversa. Gli operatori sono elencati nella tabella in ordine di priorità, dalla maggiore alla minore:

Operatore	Operazione eseguita
&	AND bit a bit
^	XOR bit a bit
	OR bit a bit

Per informazioni sull'uso degli operatori bit a bit, vedere “[Uso degli operatori bit a bit](#)” a pagina 165. Per informazioni specifiche su ogni operatore bit a bit, vedere la relativa voce nella *Guida di riferimento di ActionScript 2.0*.

Uso degli operatori bit a bit

Gli operatori bit a bit elaborano internamente i numeri a virgola mobile per trasformarli in numeri interi a 32 bit. L'operazione eseguita dipende dal tipo di operatore usato, ma tutte le operazioni bit a bit valutano separatamente ogni numero binario (bit) del numero intero a 32 bit, per calcolare un altro valore. Per un elenco degli operatori di spostamento bit a bit, vedere “[Informazioni sugli operatori di spostamento bit a bit](#)” a pagina 164. Per un elenco degli operatori logici bit a bit, vedere “[Informazioni sugli operatori logici bit a bit](#)” a pagina 165.

L'uso degli operatori bit a bit in Flash non è molto comune, ma in alcune circostanze può risultare utile, ad esempio se si desidera creare una matrice di autorizzazioni per un progetto Flash senza creare variabili separate per ogni tipo di autorizzazione. In questo caso è possibile ricorrere agli operatori bit a bit.

L'esempio seguente mostra come utilizzare l'operatore OR bit a bit con il metodo `Array.sort()` per specificare le opzioni di ordinamento.

Per utilizzare l'operatore OR bit a bit:

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

2. Immettere il codice ActionScript seguente nel pannello Azioni:

```
var myArr:Array = new Array("Bob", "Dan", "doug", "bill", "Hank",
    "tom");
trace(myArr); // Bob,Dan,doug,bill,Hank,tom
myArr.sort(Array.CASEINSENSITIVE | Array.DESCENDING);
trace(myArr); // tom,Hank,doug,Dan,Bob,bill
```

La prima riga definisce un array di nomi casuali, li traccia e li visualizza nel pannello Output. In seguito viene chiamato il metodo `Array.sort()` e vengono specificate due opzioni di ordinamento tramite i valori costanti `Array.CASEINSENSITIVE` e `Array.DESCENDING`. Il risultato del metodo di ordinamento causa un ordinamento decrescente (dalla z alla a) degli elementi dell'array. Nella ricerca non viene fatta distinzione tra maiuscole e minuscole e pertanto a e A sono considerate identiche. In una ricerca con distinzione tra maiuscole e minuscole, Z verrebbe prima di a.

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript. Il testo seguente viene visualizzato nel pannello Output:

```
Bob,Dan,doug,bill,Hank,tom
tom,Hank,doug,Dan,Bob,bill
```

Nel metodo di ordinamento sono disponibili cinque opzioni:

- 1 o `Array.CASEINSENSITIVE` (binary = 1)
- 2 o `Array.DESCENDING` (binary = 10)
- 4 o `Array.UNIQUESORT` (binary = 100)
- 8 o `Array.RETURNINDEXEDARRAY` (binary = 1000)
- 16 o `Array.NUMERIC` (binary = 10000)

Le opzioni di ordinamento per un array possono essere definite in tre modi diversi:

```
my_array.sort(Array.CASEINSENSITIVE | Array.DESCENDING); // Costanti
my_array.sort(1 | 2); // Numeri
my_array.sort(3); // Somma dei numeri
```

Sebbene possa non essere subito evidente, i valori numerici delle opzioni di ordinamento sono in realtà cifre bit a bit (binarie o in base 2). Il valore costante `Array.CASEINSENSITIVE` è uguale al valore numerico 1 che corrisponde anche al valore binario 1. Il valore costante `Array.DESCENDING` corrisponde al valore numerico 2 o al valore binario 10.

L'uso di numeri binari può creare confusione. Nel formato binario sono possibili solo due valori, 1 o 0, pertanto il valore 2 è rappresentato da 10. Per rappresentare il numero 3 in formato binario, è necessario utilizzare 11 (1+10). Il numero 4 in formato binario è 100, il 5 è 101 e così via.

Il codice ActionScript seguente illustra come ordinare un array di valori numerici in ordine decrescente utilizzando l'operatore AND bit a bit per sommare le costanti `Array.DESCENDING` e `Array.NUMERIC`.

```
var scores:Array = new Array(100,40,20,202,1,198);
trace(scores); // 100,40,20,202,1,198
trace(scores.sort()); // 1,100,198,20,202,40
var flags:Number = Array.NUMERIC|Array.DESCENDING;
trace(flags); // 18 (base 10)
trace(flags.toString(2)); // 10010 (binario -- base 2)
trace(scores.sort(flags)); // 202,198,100,40,20,1
```

Informazioni sull'operatore condizionale

L'operatore condizionale è un operatore ternario, ovvero accetta tre operandi. Può essere utilizzato come metodo rapido per applicare l'istruzione condizionale `if...else`:

Operatore	Operazione eseguita
------------------	----------------------------

<code>?:</code>	Condizionale
-----------------	--------------

Per informazioni sull'uso dell'operatore condizionale e per un esempio, vedere “[Informazioni sull'operatore condizionale e sintassi alternativa](#)” a pagina 118.

Uso degli operatori in un documento

Nell'esempio seguente, il metodo `Math.round()` consente di arrotondare i calcoli a un numero arbitrario di cifre decimali. Questo metodo arrotonda il valore del parametro `x` all'intero più prossimo per eccesso o per difetto e restituisce il valore. Modificando leggermente il codice, è possibile arrotondare i numeri in Flash a un determinato numero di cifre decimali.

Nell'esempio seguente vengono utilizzati anche gli operatori di divisione e moltiplicazione per calcolare il punteggio di un utente in base al numero di risposte corrette diviso per il numero di domande a cui è stata fornita risposta. Il punteggio dell'utente può anche essere moltiplicato per un numero per essere visualizzato in formato percentuale, da 0% a 100%. Di seguito viene utilizzato l'operatore di addizione per concatenare il punteggio dell'utente in una stringa visualizzata nel pannello Output.

Per utilizzare gli operatori in ActionScript:

1. Creare un nuovo documento Flash.
 2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale principale:
- ```
var correctAnswers:Number = 11;
var totalQuestions:Number = 13;
// Arrotonda al numero intero più prossimo
//var score:Number = Math.round(correctAnswers / totalQuestions * 100);
// Arrotonda a due cifre decimali
var score:Number = Math.round(correctAnswers / totalQuestions * 100 * 100) / 100;
trace("You got " + correctAnswers + " out of " + totalQuestions + " answers correct, for a score of " + score + "%.");
```

3. Selezionare Controllo > Prova filmato.

Il pannello Output visualizza:

You got 11 out of 13 answers correct, for a score of 84.62%.

Quando nell'esempio si chiama `Math.round()`, il punteggio viene arrotondato al numero intero più prossimo (85) e visualizzato nel pannello Output. Se si moltiplica il numero per 100 prima di chiamare `Math.round()` e quindi lo si divide per 100, è possibile eseguire l'arrotondamento a due cifre decimali per ottenere un punteggio più preciso.

4. Provare a modificare la variabile `correctAnswers` a 3 e selezionare Controllo > Prova filmato per provare nuovamente il file SWF.

Se ad esempio si crea un'applicazione per valutare la preparazione degli utenti, può essere utile creare una serie di domande a scelta multipla o con risposta vero/falso, utilizzando i componenti RadioButton e Label. Quando l'utente, dopo aver risposto alle domande, fa clic sul pulsante di invio, è possibile confrontare le risposte con una chiave e quindi calcolare il punteggio.

## CAPITOLO 5

# Funzioni e metodi

Comprendere le funzioni è importante quando si scrivono script di ActionScript, si creano classi e si utilizzano metodi. I tipi di funzioni disponibili sono diversi. In questo capitolo verranno fornite informazioni sulle funzioni e sui metodi: come scriverli e come utilizzarli nelle applicazioni quando si usano classi incorporate. Nel [Capitolo 6, “Classi”](#) è illustrata la creazione di classi personalizzate nelle quali è necessario la scrittura di funzioni. Si apprenderà anche come scrivere funzioni nei file di classe ActionScript.

Queste classi possono essere utilizzate nel codice per aggiungere interattività, animazioni e altri effetti alle applicazioni. In questo capitolo vengono illustrati i tipi di funzioni che è possibile scrivere nelle applicazioni Flash. Per ulteriori informazioni sul significato delle funzioni e dei metodi e per esaminare esercizi nei quali si scrivono e si utilizzano funzioni e metodi in Flash, consultare i seguenti argomenti:

|                                                         |     |
|---------------------------------------------------------|-----|
| <a href="#">Informazioni su funzioni e metodi</a> ..... | 169 |
| <a href="#">Nozioni fondamentali sui metodi</a> .....   | 192 |

## Informazioni su funzioni e metodi

I metodi e le funzioni sono blocchi di codice ActionScript riutilizzabili in qualsiasi punto di un file SWF. È possibile creare una funzione nel file FLA o in un file ActionScript esterno e quindi chiamarla da qualunque punto dei documenti. I metodi sono semplicemente funzioni che si trovano all'interno di una definizione di classe ActionScript. È possibile definire funzioni per eseguire una serie di istruzioni sui valori specificati. Le funzioni possono anche restituire altri valori. Una volta definita una funzione, è possibile chiamarla da qualsiasi linea temporale, inclusa la linea temporale di un file SWF caricato.

Se a una funzione vengono passati valori come parametri, la funzione può eseguire calcoli utilizzando i valori forniti. Ogni funzione presenta caratteristiche proprie e alcune funzioni richiedono il passaggio di un determinato tipo o numero di valori. Se viene passato un numero di parametri superiore a quello richiesto dalla funzione, i valori aggiuntivi vengono ignorati. Se un parametro obbligatorio non viene passato, la funzione assegna ai parametri vuoti il tipo di dati `undefined`. Questa situazione può causare errori in fase di runtime. Una funzione può inoltre restituire altri valori (vedere “[Restituzione di valori da funzioni a pagina 190](#)”).

**NOTA**

È possibile chiamare solo funzioni che si trovano in un fotogramma già raggiunto dall'indicatore di riproduzione.

Una funzione scritta correttamente può essere paragonata a una scatola nera. Se la funzione contiene commenti adeguati relativamente ai parametri da passare, ai dati restituiti e allo scopo, l'utente che la utilizza non ha la necessità di comprenderne esattamente il funzionamento interno.

La sintassi di base di una semplice *funzione con nome* è:

```
function traceMe() {
 trace("messaggio");
}
traceMe();
```

Per informazioni sulla scrittura di funzioni con nome, vedere “[Scrittura di funzioni con nome a pagina 175](#)”.

La sintassi di base di una semplice funzione con nome basata sull'esempio precedente passando un parametro, `yourMessage`, è:

```
function traceMe(yourMessage:String) {
 trace(yourMessage);
}
traceMe("Come va?");
```

In alternativa, se si desidera passare più parametri, è possibile utilizzare il codice seguente:

```
var yourName:String = "Ester";
var yourAge:String = "65";
var favSoftware:String = "Flash";
function traceMe(favSoftware:String, yourName:String, yourAge:String) {
 trace("Sono " + yourName + ", apprezzo " + favSoftware + " e ho " +
 yourAge + " anni.");
}
traceMe(favSoftware,yourName,yourAge);
```

Per ulteriori informazioni sul passaggio di parametri, vedere “[Passaggio di parametri a una funzione a pagina 188](#)”.

È possibile scrivere numerosi tipi di funzioni. Per ulteriori informazioni su come scrivere le funzioni, come pure per collegamenti alle sezioni relative alla scrittura di tipi specifici di funzioni, vedere “[Informazioni sui metodi e sulle funzioni](#)” a pagina 171. Per un esempio che confronti metodi e funzioni, vedere “[Nozioni fondamentali sui metodi](#)” a pagina 192.

**NOTA**

Per ulteriori informazioni su come scrivere il codice utilizzando Assistente script, vedere “[Uso di Assistente per scrivere codice ActionScript](#)” a pagina 374, “[Creazione di un evento startDrag/stopDrag mediante Assistente script](#)” a pagina 378 e l'esercitazione ActionScript:Uso della modalità Assistente script (che inizia con “[Apertura del documento di inizio](#)” a pagina 225).

Per ulteriori informazioni sulle funzioni e sui metodi, consultare i seguenti argomenti:

- “[Informazioni sui metodi e sulle funzioni](#)” a pagina 171

## Informazioni sui metodi e sulle funzioni

Le funzioni che appartengono a una classe sono detti *metodi* della classe. Nelle applicazioni è possibile utilizzare molti tipi di funzioni, comprese le funzioni incorporate, le funzioni con nome e quelle definite dall'utente, le funzioni anonime, le funzioni di callback, le funzioni di costruzione e le funzioni letterali. Nelle sezioni riportate di seguito sono fornite informazioni su come definire queste funzioni.

È anche possibile scrivere funzioni in un file di classe ActionScript. Negli script si utilizzano queste funzioni che operano come metodi. Nell'esempio seguente, la classe Person comprende un metodo della funzione di costruzione, metodi di classe, metodi di istanza e metodi supplementari (getter e setter). I commenti in questo codice di esempio evidenziano la posizione di questi metodi nel codice.

**NOTA**

Per ulteriori informazioni su come scrivere file di classe, come i seguenti, vedere [Capitolo 6, “Classi” a pagina 197](#).

```
class Person {
 public static var numPeople:Number = 0;

 // Membri di istanza
 private var _speed:Number;

 // funzione di costruzione
 public function Person(speed:Number) {
 Person.numPeople++;
 this._speed = speed;
 }

 // Metodi statici
```

```

public static function getPeople():Number {
 return Person.numPeople;
}

// Metodi di istanza
public function walk(speed:Number):Void {
 this._speed = speed;
}
public function run():Void {
 this._speed *= 2;
}
public function rest():Void {
 this._speed = 0;
}

// Getter/setter (metodi supplementari)
public function get speed():Number {
 return this._speed;
}
}

```

Per una dimostrazione completa su come scrivere metodi simili al precedente, vedere [Capitolo 6, “Classi” a pagina 197](#). I metodi utilizzati nel codice possono appartenere a una classe incorporata nel linguaggio ActionScript. MovieClip e Math sono esempi di classi di primo livello che possono essere utilizzate in un'applicazione. Quando si utilizzano nel codice i metodi di queste classi, sono funzioni scritte nella classe incorporata (simile al precedente esempio di codice). In alternativa, è possibile utilizzare i metodi di una classe personalizzata scritta appositamente.

Le funzioni che non appartengono a una classe vengono chiamate *funzioni di primo livello* (talvolta chiamate *funzioni predefinite* o *incorporate*), in quanto possono essere chiamate senza una funzione di costruzione. Esempi di funzioni incorporate nel primo livello del linguaggio ActionScript sono `trace()` e `setInterval()`.

Per aggiungere una funzione di primo livello al proprio codice è sufficiente aggiungere una sola riga nel riquadro Script del pannello Azioni. Immettere ad esempio il seguente codice:

```
trace("messaggio");
```

Quando si prova il file SWF con questa unica riga di codice, viene chiamata la funzione di primo livello `trace()` e nel pannello Output viene visualizzato il testo.

Si ricordi che: quando si desidera assegnare un metodo a una proprietà, si omettono le parentesi dopo il nome del metodo poiché si sta passando un riferimento alla funzione:

```
my_mc.myMethod = aFunction;
```

Per richiamare un metodo nel codice, è invece necessario aggiungere le parentesi dopo il nome del metodo:

```
my_mc.myMethod();
```



Per ulteriori informazioni sulle funzioni di primo livello, vedere “[Informazioni sulle funzioni incorporate e di primo livello](#)” a pagina 173.

Le funzioni possono essere definite anche in diversi altri modi. Per ulteriori informazioni su ciascun tipo di funzione, consultare le seguenti sezioni:

- “[Informazioni sulle funzioni incorporate e di primo livello](#)” a pagina 173
- “[Scrittura di funzioni con nome](#)” a pagina 175
- “[Scrittura di funzioni anonime e di callback](#)” a pagina 176
- “[Informazioni sui valori letterali di funzione](#)” a pagina 179
- “[Identificazione e chiamata delle funzioni definite dall'utente](#)” a pagina 181
- “[Informazioni sulle funzioni di costruzione](#)” a pagina 179

Per informazioni sulla scrittura e l'uso di funzioni e metodi, consultare le seguenti sezioni correlate. Per informazioni sull'uso delle funzioni, vedere “[Uso delle funzioni in Flash](#)” a pagina 183. Per informazioni sull'uso dei metodi, vedere “[Nozioni fondamentali sui metodi](#)” a pagina 192.



Per ulteriori informazioni su come scrivere il codice utilizzando Assistente script, vedere “[Uso di Assistente per scrivere codice ActionScript](#)” a pagina 374, “[Creazione di un evento startDrag/stopDrag mediante Assistente script](#)” a pagina 378 e l'esercitazione ActionScript:Uso della modalità Assistente script (che inizia con “[Apertura del documento di inizio](#)” a pagina 225).

## Informazioni sulle funzioni incorporate e di primo livello

Come illustrato in “[Informazioni su funzioni e metodi](#)” a pagina 169, una funzione è un blocco di codice ActionScript riutilizzabile in qualsiasi punto di un file SWF. Se a una funzione vengono passati dei valori come parametri, la funzione opera su tali valori. Una funzione può inoltre restituire altri valori.

È possibile utilizzare le funzioni incorporate nel linguaggio ActionScript. Si può trattare di funzioni di primo livello, come descritto in “[Informazioni sui metodi e sulle funzioni](#)” a pagina 171 o la funzione può trovarsi in una classe incorporata, quale ad esempio Math o MovieClip, che può essere utilizzata come un metodo nell'applicazione.

In ActionScript si utilizzano le funzioni incorporate per eseguire determinate attività e per accedere alle informazioni. È possibile, ad esempio, ottenere il tempo di riproduzione di un file SWF in numero di millisecondi utilizzando `getTimer()` o il numero di versione di Flash Player in cui è caricato il file utilizzando `getVersion()`. Le funzioni appartenenti a un oggetto sono denominate *metodi*. Quelle che non appartengono a un oggetto sono denominate *funzioni -di primo livello* e sono contenute in sottocategorie della categoria Funzioni globali disponibile nel pannello Azioni.

Alcune funzioni incorporate richiedono il passaggio di determinati valori. Se si passa un numero di parametri superiore a quello richiesto dalla funzione, i valori aggiuntivi vengono ignorati. Se non si passa un parametro obbligatorio, ai parametri vuoti viene assegnato il tipo di dati `undefined`. Questo comportamento può generare errori in fase di runtime.

**NOTA**

È possibile chiamare solo funzioni che si trovano in un fotogramma già raggiunto dall'indicatore di riproduzione.

Le funzioni di primo livello sono di facile utilizzo. Per chiamare una funzione, è sufficiente utilizzarne il nome e passare tutti i parametri richiesti. (Per informazioni su parametri richiesti, vedere la voce relativa alla funzione nella *Guida di riferimento di ActionScript 2.0*). Ad esempio, aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
trace("messaggio");

function myTimer():Void {
 trace(getTimer());
}
var intervalID:Number = setInterval(myTimer, 100);
```

Questo codice crea un semplice timer con `getTimer()` e utilizza le funzioni di primo livello `setInterval()` e `trace()` per visualizzare il numero di millisecondi trascorso dall'inizio della riproduzione del file SWF in Flash Player.

La chiamata di una funzione di primo livello è analoga a quella di una *funzione definita dall'utente*. Per ulteriori informazioni, vedere “[Identificazione e chiamata delle funzioni definite dall'utente](#)” a pagina 181. Per ulteriori informazioni sulle singole funzioni, vedere le relative voci nella *Guida di riferimento di ActionScript 2.0*.

## Scrittura di funzioni con nome

Una funzione con nome è un tipo di funzione che viene comunemente creata nel codice ActionScript per eseguire tutti i tipi di azioni. Quando si crea un file SWF, le funzioni con nome sono compilate per prime; cioè, è possibile fare riferimento a queste funzioni in qualsiasi punto del codice, a condizione che la funzione sia stata definita nel fotogramma corrente o precedente. Ad esempio, se una funzione è definita nel fotogramma 2 di una linea temporale, non sarà possibile accedervi dal fotogramma 1 della linea temporale.

Il formato standard di una funzione con nome è il seguente:

```
function Nome funzione(parametri) {
 // Blocco di funzione
}
```

Questa porzione di codice comprende le parti seguenti:

- Nomefunzione è il nome univoco della funzione. Tutti i nomi di funzione in un documento devono essere univoci.
- parametri contiene uno o più parametri che vengono passati alla funzione. I parametri sono detti anche *argomenti*. Per ulteriori informazioni sui parametri, vedere “[Passaggio di parametri a una funzione](#)” a pagina 188.
- // Blocco di funzione contiene tutto il codice ActionScript relativo alla funzione. Questa parte contiene le istruzioni che eseguono le azioni, ovvero il codice che si desidera eseguire. Il commento // Blocco di funzione è un segnaposto che indica dove deve essere inserito il blocco della funzione.

### Per utilizzare una funzione con nome:

1. Creare un nuovo documento denominato **namedFunc.fla**.
2. Importare un breve file audio nella libreria selezionando File > Importa > Importa nella libreria e selezionando un file audio.
3. Fare clic con il pulsante destro del mouse sul file audio e selezionare Concatenamento.
4. Digitare **mySoundID** nella casella di testo Identificatore.
5. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
function myMessage() {
 trace("mySoundID completato");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = myMessage;
my_sound.start();
```

In questo codice si crea una funzione con nome denominata `myMessage`, che verrà utilizzata in un momento successivo per chiamare una funzione `trace()`.

**6.** Selezionare Controllo > Prova filmato per provare il file SWF.

Si utilizza l'istruzione `funzione` per creare la propria funzione in ActionScript. I parametri sono opzionali, ma è necessario includere le parentesi quadre anche se non vengono specificati. Il contenuto tra le parentesi graffe (`{}`) è detto *blocco di funzione*.

Le funzioni possono essere scritte sulla linea temporale principale o all'interno di file ActionScript esterni, ad esempio file di classe.

Anche le funzioni di costruzione nei file di classe utilizzano lo stesso formato, anche se il nome della funzione corrisponde alla classe. Per ulteriori informazioni sulle funzioni di costruzione, vedere “[Creazione della funzione di costruzione](#)” a pagina 243. Per informazioni ed esempi sulla scrittura di funzioni nelle classi, consultare anche il [Capitolo 6, “Classi”](#) a pagina 197.

## Scrittura di funzioni anonime e di callback

Una funzione con nome è una funzione alla quale si fa riferimento in uno script prima o dopo la definizione, mentre una *funzione anonima* è una funzione senza nome che fa riferimento a se stessa; si fa riferimento alla funzione anonima quando la si crea. Nella scrittura del codice ActionScript, verranno create numerose funzioni anonime.

In genere si ricorre a queste funzioni quando si utilizzano gestori di eventi. Per scrivere una funzione anonima, si memorizza un valore letterale di funzione all'interno di una variabile. In questo modo è possibile fare riferimento alla funzione all'interno del codice. L'esempio seguente illustra come scrivere una funzione anonima.

### Per scrivere una funzione anonima:

1. Creare un clip filmato sullo stage e selezionarlo.
2. Nella finestra di ispezione Proprietà digitare `my_mc` nella casella di testo Nome istanza.
3. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var myWidth = function () {
 trace(my_mc._width);
};
// In seguito è possibile aggiungere nel codice
myWidth();
```

4. Selezionare Controllo > Prova filmato.

La larghezza del clip filmato viene visualizzata nel pannello Output.

È anche possibile creare una funzione all'interno di un oggetto, ad esempio un'istanza XML o LoadVars, associando una funzione anonima a un determinato evento per creare una *funzione di callback*. Una funzione chiama una funzione di callback dopo che si verifica un evento specifico, ad esempio dopo il termine del caricamento (`onLoad()`) o dell'animazione di un oggetto (`onMotionFinished()`).

A volte è necessario, ad esempio, scrivere codice ActionScript per gestire i dati che vengono caricati in un file SWF dal server. Al termine del caricamento dei dati in un file SWF, è possibile accedere ai dati da quella posizione. È importante utilizzare codice ActionScript per verificare se i dati sono stati caricati completamente. È possibile utilizzare funzioni di callback per indicare che i dati sono stati caricati nel documento.

Nella funzione di callback seguente in cui si carica un documento XML remoto, una funzione anonima viene associata all'evento `onLoad()`. Come si può notare, nell'esempio seguente vengono utilizzati `XML.load()` e la funzione di callback. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean):Void {
 trace(success);
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

Nel frammento di codice riportato sopra si può notare che il gestore di eventi `onLoad()` utilizza una funzione anonima per gestire l'evento `onLoad()`.

Per ulteriori informazioni sulle funzioni di callback, vedere [Capitolo 9, “Gestione degli eventi” a pagina 311](#).

È anche possibile utilizzare funzioni anonime con la funzione `setInterval()`, come illustrato nel codice seguente, che utilizza `setInterval()` per chiamare la funzione anonima approssimativamente ogni 1000 millisecondi (1 secondo):

```
setInterval(function() {trace("intervallo");}, 1000);
```

Anziché funzioni anonime, è possibile utilizzare funzioni con nome. Le funzioni con nome sono spesso più facili da leggere e da comprendere (tranne che in alcune situazioni, come per le funzioni di callback). È possibile fare riferimento a una funzione con nome che si trova in un punto successivo del codice, ovvero prima che esista sulla linea temporale.

Non è possibile fare riferimento alle funzioni anonime da qualunque punto del codice (a meno che non venga assegnata a una variabile), come nel caso delle funzioni con nome. Si supponga di disporre di funzioni anonime nel fotogramma 5 del file FLA, come nell'esempio seguente:

```
//con un clip filmato denominato my_mc che estende la linea temporale
stop();
var myWidth = function () {
 trace(my_mc._width);
};
```

Il seguente codice inserito nel fotogramma 1 non potrà fare riferimento alla funzione:  
`myWidth();`

Analogamente, il seguente codice inserito in un fotogramma qualsiasi non funzionerà:

```
myWidth();
var myWidth:Function = function () {
 trace(my_mc._width);
};
```

Tuttavia, questo codice funziona correttamente:

```
var myWidth:Function = function () {
 trace(my_mc._width);
};
myWidth();
```



È anche possibile collocare `myWidth()` in qualsiasi fotogramma successivo a quello che contiene la funzione `myWidth`.

Quando si definisce una funzione con nome, la chiamata in uno script di fotogrammi funziona, al contrario di un equivalente codice con una funzione anonima:

```
// il codice seguente funziona poiché si sta chiamando una funzione con
// nome:
myWidth();
function myWidth() {
 trace("foo");
}

// il codice seguente non funziona poiché si sta chiamando una funzione
// anonima:
myWidth();
var myWidth:Function = function () {
 trace("foo");
};
```

Per ulteriori informazioni, vedere “[Scrittura di funzioni con nome](#)” a pagina 175.

**NOTA**

Per ulteriori informazioni su come scrivere il codice utilizzando Assistente script, vedere “[Uso di Assistente per scrivere codice ActionScript](#)” a pagina 374, “[Creazione di un evento startDrag/stopDrag mediante Assistente script](#)” a pagina 378 e l'esercitazione ActionScript:Uso della modalità Assistente script (che inizia con “[Apertura del documento di inizio](#)” a pagina 225).

## Informazioni sui valori letterali di funzione

Per *valore letterale di funzione* si intende una funzione senza nome dichiarata in un'espressione anziché in un'istruzione. I valori letterali di funzione consentono di utilizzare una funzione temporaneamente o al posto di un'espressione. La sintassi è la seguente:

```
function (param1, param2, ecc.) {
 // istruzioni
};
```

Ad esempio, il codice seguente utilizza una funzione letterale come un'espressione:

```
var yourName:String = "Ester";
setInterval(function() {trace(yourName);}, 200);
```

**NOTA**

Quando si ridefinisce una funzione letterale, la nuova definizione della funzione sostituisce quella esistente.

Un valore letterale di funzione può essere memorizzato in una variabile per potervi accedere in un punto successivo del codice tramite una *funzione anonima*. Per ulteriori informazioni, vedere “[Scrittura di funzioni anonime e di callback](#)” a pagina 176.

## Informazioni sulle funzioni di costruzione

La funzione di costruzione di una classe è una funzione speciale che viene chiamata automaticamente quando si crea un'istanza di una classe tramite la parola chiave new (ad esempio `my_xml:XML = new XML();`) e presenta lo stesso nome della classe che la contiene. La classe personalizzata Person creata precedentemente, ad esempio, conteneva la seguente funzione di costruzione:

```
public function Person(speed:Number) {
 Person.numPeople++;
 this._speed = speed;
}
```

Quindi è possibile creare una nuova istanza con:

```
var myPerson:Person = new Person();
```



Se non si dichiara esplicitamente alcuna funzione di costruzione, ossia non viene creata una funzione il cui nome corrisponde a quello della classe, il compilatore crea automaticamente una funzione di costruzione vuota.

Una classe può contenere solo una funzione di costruzione; in ActionScript 2.0 le funzioni di costruzione multiple non sono consentite. Inoltre, una funzione di costruzione non può avere un tipo restituito. Per ulteriori informazioni sulla scrittura di funzioni in un file di classe, consultare “[Creazione della funzione di costruzione](#)” a pagina 243.

## Definizione di funzioni globali e associate alla linea temporale

In “[Informazioni su funzioni e metodi](#)” a pagina 169 sono stati presentati i diversi tipi di funzioni disponibili in Flash. Analogamente alle variabili, le funzioni sono associate alla linea temporale del clip filmato che le definisce. È quindi necessario utilizzare un percorso target per chiamarle. Come nel caso delle variabili, è possibile utilizzare l'identificatore `_global` per dichiarare una funzione globale che sia disponibile per tutte le linee temporali e le aree di validità senza l'uso di un percorso target. Per definire una funzione globale, è necessario far precedere il nome di tale funzione dall'identificatore `_global`, come nell'esempio seguente:

```
_global.myFunction = function(myNum:Number):Number {
 return (myNum * 2) + 3;
};
trace(myFunction(5)) // 13
```

Per informazioni su `_global` e l'area di validità, vedere “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362.

Per definire una funzione della linea temporale, utilizzare l'istruzione `function` seguita dal nome della funzione, dai parametri da passare alla funzione e dalle istruzioni ActionScript che indicano le operazioni da essa eseguite.

Nell'esempio seguente viene definita una funzione denominata `areaOfCircle` con il parametro `radius`:

```
function areaOfCircle(radius:Number):Number {
 return (Math.PI * radius * radius);
}
trace (areaOfCircle(8));
```

Le funzioni possono essere definite anche in diversi altri modi. Per ulteriori informazioni su ciascun tipo di funzione, consultare le seguenti sezioni:

- “[Informazioni sulle funzioni incorporate e di primo livello](#)” a pagina 173
- “[Scrittura di funzioni con nome](#)” a pagina 175
- “[Scrittura di funzioni anonime e di callback](#)” a pagina 176
- “[Informazioni sui valori letterali di funzione](#)” a pagina 179
- “[Informazioni sulle funzioni di costruzione](#)” a pagina 179
- “[Identificazione e chiamata delle funzioni definite dall'utente](#)” a pagina 181

Per informazioni sull'assegnazione di nomi alle funzioni, vedere “[Assegnazione di nomi alle funzioni](#)” a pagina 183. Per un esempio dettagliato dell'uso delle funzioni in un file di classe esterno, vedere “[Uso delle funzioni in Flash](#)” a pagina 183 e [Capitolo 6, “Classi”](#) a pagina 197.

**NOTA**

Per ulteriori informazioni su come scrivere il codice utilizzando Assistente script, vedere “[Uso di Assistente per scrivere codice ActionScript](#)” a pagina 374, “[Creazione di un evento startDrag/stopDrag mediante Assistente script](#)” a pagina 378 e l'esercitazione ActionScript:Uso della modalità Assistente script (che inizia con “[Apertura del documento di inizio](#)” a pagina 225).

## Identificazione e chiamata delle funzioni definite dall'utente

Per *funzioni definite dall'utente* si intendono semplici funzioni create dall'utente da utilizzare nelle applicazioni. Si contrappongono alle funzioni delle classi incorporate che invece eseguono operazioni predefinite. Il nome della funzione viene assegnato dal programmatore che aggiunge quindi le istruzioni all'interno del blocco della funzione. Le sezioni precedenti illustrano la scrittura di funzioni con nome, senza nome e di callback. Per informazioni sull'assegnazione di nomi alle funzioni, vedere “[Assegnazione di nomi alle funzioni](#)” a pagina 183. Per informazioni sull'utilizzo delle funzioni, vedere “[Uso delle funzioni in Flash](#)” a pagina 183.

È possibile usare un percorso target per chiamare una funzione in qualsiasi linea temporale e da qualsiasi linea temporale, inclusa quella di un file SWF caricato. Per chiamare una funzione, specificare il percorso target includendo il nome della funzione, se necessario, e passare tutti i parametri richiesti all'interno di parentesi. Per le funzioni definite dall'utente è possibile adottare diversi formati di sintassi. Nell'esempio di codice seguente viene utilizzato un percorso per chiamare la funzione `initialize()`, che viene definita sulla linea temporale principale corrente e non richiede parametri:

```
this.initialize();
```

Nell'esempio seguente viene utilizzato un percorso relativo per chiamare la funzione `list()` definita nel clip filmato `functionsClip`:

```
this._parent.functionsClip.list(6);
```

Per informazioni sulla scrittura di funzioni con nome, vedere “[Scrittura di funzioni con nome](#)” a pagina 175. Per ulteriori informazioni sui parametri, vedere “[Passaggio di parametri a una funzione](#)” a pagina 188.

È anche possibile definire funzioni con nome personalizzate. La seguente funzione con nome `helloWorld()`, ad esempio, è una funzione definita dall'utente:

```
function helloWorld() {
 trace("Hello world!");
};
```

L'esempio seguente mostra come utilizzare una funzione definita dall'utente in un file FLA.

#### **Per creare e chiamare un semplice funzione definita dall'utente:**

1. Creare un nuovo documento Flash e salvarlo come **udf.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
function traceHello(name:String):Void {
 trace("hello, " + name + "!");
}
traceHello("world"); // hello, world!
```

Il codice precedente crea una funzione definita dall'utente denominata `traceHello()` che accetta un solo argomento, `name`, e visualizza un messaggio di benvenuto. Per chiamare la funzione definita dall'utente, è possibile chiamare `traceHello` dalla stessa linea temporale della definizione della funzione e passare un unico valore stringa.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Per ulteriori informazioni sulle funzioni con nome, vedere “[Scrittura di funzioni con nome](#)” a pagina 175. Le classi contengono molte funzioni definite dall'utente. Per informazioni sulla scrittura di funzioni all'interno di file di classe, vedere “[Uso delle funzioni in Flash](#)” a pagina 183. Consultare inoltre le seguenti sezioni nel Capitolo 6, “Classi”: “[Uso di metodi e proprietà da un file di classe](#)” a pagina 218, “[Informazioni su metodi e proprietà \(membri\) pubblici, privati e statici](#)” a pagina 220 e “[Informazioni sui membri di classe](#)” a pagina 224.

## Assegnazione di nomi alle funzioni

I nomi di funzione devono iniziare con una lettera minuscola e descrivere il valore restituito dalla funzione, se presente. Se, ad esempio, la funzione restituisce il titolo di un brano, è possibile denominare la funzione `getBranoCorrente()`.

Stabilire uno standard per raggruppare funzioni simili (funzioni correlate in base alla funzionalità) perché ActionScript non consente l'overload. Nella programmazione orientata agli oggetti (OOP, Object Oriented Programming), per *overload* si intende la possibilità che le funzioni personalizzate si comportino in modo diverso a seconda del tipo di dati che viene loro passato.

Come con le variabili, non è possibile utilizzare caratteri speciali e i nomi dei metodi non possono iniziare con una cifra. Per ulteriori informazioni, vedere “[Convenzioni di denominazione](#)” a pagina 793. Per informazioni sull'assegnazione di nomi ai metodi, vedere “[Assegnazione di nomi ai metodi](#)” a pagina 195.

## Uso delle funzioni in Flash

Questa sezione illustra come utilizzare funzioni in un'applicazione. In alcuni degli esempi di codice seguenti viene utilizzato codice ActionScript contenuto nel file FLA, mentre per altri esempi di codice le funzioni sono inserite in un file di classe per consentire di eseguire un confronto. Per ulteriori informazioni ed esempi sull'utilizzo delle funzioni in un file di classe, vedere [Capitolo 6, “Classi” a pagina 197](#). Per informazioni dettagliate e spiegazioni su come scrivere funzioni per un file di classe, vedere “[Esempio: creazione di classi personalizzate](#)” a pagina 237.

Per ridurre la quantità di codice da scrivere e anche le dimensioni del file SWF, cercare di riutilizzare i blocchi di codice il più possibile, ad esempio chiamando una funzione più volte anziché creare porzioni diverse di codice ogni volta. Le funzioni possono essere rappresentate da porzioni di codice generiche, pertanto è possibile utilizzare gli stessi blocchi di codice per scopi leggermente diversi in un file SWF. Riutilizzando il codice è possibile creare applicazioni efficienti, riducendo la quantità di codice ActionScript da scrivere e di conseguenza il tempo di sviluppo.

È possibile creare funzioni in un file FLA, in un file di classe o scrivere codice ActionScript in un componente basato su codice. L'esempio seguente illustra come creare funzioni sulla linea temporale e in un file di classe.

**SUGGERIMENTO**

Inserendo il codice in file di classe o componenti basati su codice, è possibile condividere, distribuire o riutilizzare blocchi di codice in modo semplice. Gli utenti possono installare il componente, trascinarlo nello stage e utilizzare il codice memorizzato nel file, come flusso di lavoro per componenti basati su codice disponibile in Flash (Window > Librerie comuni > Classi).

L'esempio seguente mostra come creare e chiamare una funzione in un file FLA.

**Per creare e chiamare una funzione in un file FLA:**

1. Creare un nuovo documento Flash e salvarlo come **basicFunction.fla**.
2. Selezionare Finestra > Azioni per aprire il pannello Azioni.
3. Immettere il codice ActionScript seguente nel pannello Script:

```
function helloWorld() {
 // Istruzioni
 trace("Hello world!");
};
```

Questo codice ActionScript dichiara la funzione (definita dall'utente e con nome) `helloWorld()`. Se si prova il file SWF in questo punto, non si ottiene alcun risultato. Nel pannello Output non viene, ad esempio, visualizzata l'istruzione `trace`. Per visualizzare l'istruzione `trace` occorre chiamare la funzione `helloWorld()`.

4. Digitare il seguente codice ActionScript dopo la funzione:

```
helloWorld();
```

Questo codice chiama la funzione `helloWorld()`.

5. Selezionare Controllo > Prova filmato per provare il file FLA.

Il testo seguente viene visualizzato nel pannello Output: Hello world!

Per informazioni sul passaggio di valori (parametri) a una funzione, vedere “[Passaggio di parametri a una funzione](#)” a pagina 188.

Le funzioni possono essere scritte sulla linea temporale principale in modi diversi. In particolare, è possibile utilizzare funzioni anonime e funzioni con nome, ad esempio tramite la sintassi seguente:

```
function myCircle(radius:Number):Number {
 return (Math.PI * radius * radius);
}
trace(myCircle(5));
```

Le funzioni anonime sono spesso più difficili da leggere. Confrontare il codice seguente con il precedente.

```
var myCircle:Function = function(radius:Number):Number {
 // Blocco di funzione
 return (Math.PI * radius * radius);
};
trace(myCircle(5));
```

se si utilizza ActionScript 2.0 possono anche essere definite in file di classe, come illustrato nell'esempio seguente:

```
class Circle {
 public function area(radius:Number):Number {
 return (Math.PI * Math.pow(radius, 2));
 }
 public function perimeter(radius:Number):Number {
 return (2 * Math.PI * radius);
 }
 public function diameter(radius:Number):Number {
 return (radius * 2);
 }
}
```

Per ulteriori informazioni sulla scrittura di funzioni in un file di classe, consultare il [Capitolo 6, “Classi” a pagina 197](#)

Come illustrato nel precedente codice di esempio, non è necessario inserire le funzioni in una linea temporale. Anche nell'esempio seguente le funzioni vengono inserite in un file di classe. Questo comportamento è consigliabile per la creazione di applicazioni di grandi dimensioni tramite ActionScript 2.0 perché consente di riutilizzare con facilità il codice in più applicazioni. Se si desidera riutilizzare le funzioni in altre applicazioni, è possibile importare la classe esistente anziché riscrivere il codice da zero oppure duplicare le funzioni nella nuova applicazione.

**Per creare funzioni in un file di classe:**

1. Creare un nuovo documento ActionScript e salvarlo come **Utils.as**.

2. Immettere il codice ActionScript seguente nel pannello Script:

```
class Utils {
 public static function randomRange(min:Number, max:Number):Number {
 if (min > max) {
 var temp:Number = min;
 min = max;
 max = temp;
 }
 return (Math.floor(Math.random() * (max - min + 1)) + min);
 }
 public static function arrayMin(num_array:Array):Number {
 if (num_array.length == 0) {
 return Number.NaN;
 }
 num_array.sort(Array.NUMERIC | Array.DESCENDING);
 var min:Number = Number(num_array.pop());
 return min;
 }
 public static function arrayMax(num_array:Array):Number {
 if (num_array.length == 0) {
 return undefined;
 }
 num_array.sort(Array.NUMERIC);
 var max:Number = Number(num_array.pop());
 return max;
 }
}
```

3. Selezionare File > Salva per salvare il file ActionScript.

4. Creare un nuovo documento Flash e salvarlo come **classFunctions.fla** nella stessa cartella in cui si trova Utils.as.

5. Selezionare Finestra > Azioni per aprire il pannello Azioni.

6. Immettere il codice ActionScript seguente nel pannello Script:

```
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("mese: " + randomMonth);
trace("min: " + min); // -3
trace("max: " + max); // 34
```

7. Selezionare Controllo > Prova filmato per provare i documenti. Il testo seguente viene visualizzato nel pannello Output:

```
messe 7
min: -3
max: 34
```

**NOTA**

Per ulteriori informazioni su come scrivere il codice utilizzando Assistente script, vedere [“Uso di Assistente per scrivere codice ActionScript” a pagina 374](#), [“Creazione di un evento startDrag/stopDrag mediante Assistente script” a pagina 378](#) e l'esercitazione ActionScript:Uso della modalità Assistente script (che inizia con [“Apertura del documento di inizio” a pagina 225](#)).

## Uso di variabili nelle funzioni

Le variabili locali sono strumenti utili per organizzare il codice e renderlo più comprensibile. Quando una funzione utilizza variabili locali, può nascondere le proprie variabili a tutti gli altri script del file SWF. Le variabili locali hanno come area di validità il corpo della funzione e cessano di esistere al termine della stessa. Qualsiasi parametro passato a una funzione viene considerato una variabile locale.

**NOTA**

In una funzione è anche possibile utilizzare variabili standard. Se, tuttavia, si modificano le variabili standard, è una buona regola di programmazione utilizzare dei commenti per segnalare tali modifiche.

### Per usare le variabili nelle funzioni:

1. Creare un nuovo documento Flash e salvarlo come **flashvariables.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var myName:String = "Ester";
var myAge:String = "65";
var myFavSoftware:String = "Flash";
function traceMe(yourFavSoftware:String, yourName:String,
 yourAge:String) {
 trace("Sono " + yourName + ", apprezzo " + yourFavSoftware + " e ho "
 + yourAge + " anni.");
}
traceMe(myFavSoftware, myName, myAge);
```

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Per ulteriori informazioni sul passaggio di parametri, vedere [“Passaggio di parametri a una funzione” a pagina 188](#). Per ulteriori informazioni su variabili e dati, consultare il Capitolo 10, [“Dati e tipi di dati” a pagina 333](#)

## Passaggio di parametri a una funzione

I parametri, detti anche *argomenti* sono gli elementi sui quali viene eseguito il codice della funzione. Nel presente manuale, i termini *parametro* e *argomento* sono equivalenti. È possibile passare parametri (valori) a una funzione e quindi utilizzare tali parametri per elaborare la funzione. I valori vengono utilizzati all'interno del blocco della funzione (istruzioni all'interno della funzione).

Talvolta i parametri sono obbligatori e talvolta sono facoltativi. In una funzione potrebbero inoltre essere presenti sia parametri obbligatori che opzionali. Se non si passa un numero sufficiente di parametri a una funzione, Flash imposta i valori dei parametri mancanti a `undefined` e si otterranno risultati imprevisti nel file SWF.

La funzione `myFunc()` seguente accetta il parametro `someText`:

```
function myFunc(someText:String):Void {
 trace(someText);
}
```

Dopo aver passato il parametro, è possibile passare un valore alla funzione quando la si chiama. Il valore viene tracciato nel pannello Output nel modo seguente:

```
myFunc("Questo è il valore tracciato");
```

Quando si chiama questa funzione, è necessario passare il numero di parametri specificato, a meno che la funzione non verifichi i valori `undefined` e imposti di conseguenza i valori predefiniti. La funzione sostituisce i valori passati ai parametri presenti nella definizione della funzione; se non sono presenti tutti i parametri, Flash ne imposta i valori a `undefined`. Nel codice ActionScript si passano spesso parametri alle funzioni.

È inoltre possibile passare più parametri a una funzione in modo semplice, come nell'esempio seguente:

```
var birthday:Date = new Date(1901, 2, 3);
trace(birthday);
```

Ogni parametro è separato da una virgola. Molte funzioni incorporate del linguaggio ActionScript dispongono di più parametri. Il metodo `startDrag()` della classe MovieClip, ad esempio, accetta cinque parametri: `lockCenter`, `left`, `top`, `right` e `bottom`:

```
startDrag(lockCenter:Boolean, left:Number, top:Number, right:Number,
bottom:Number):Void
```

### **Per passare un parametro a una funzione:**

1. Creare un nuovo documento Flash e salvarlo come **parameters.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
function traceMe(yourMessage:String):Void {
 trace(yourMessage);
}
traceMe("Come va?");
```

Le prime righe di codice creano una funzione definita dall'utente denominata `traceMe()`, che accetta un unico parametro, `yourMessage`. L'ultima riga del codice chiama la funzione `traceMe()` e passa il valore stringa "Come va?".

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

L'esempio seguente illustra come passare più parametri a una funzione.

### **Per passare più parametri a una funzione:**

1. Creare un nuovo documento Flash e salvarlo come **functionTest.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
function getArea(width:Number, height:Number):Number {
 return width * height;
}
```

La funzione `getArea` accetta due parametri: `width` e `height`.

3. Immettere il codice seguente dopo la funzione:

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

La chiamata alla funzione `getArea()` assegna i valori 10 e 12 rispettivamente ai parametri `width` e `height` e il valore restituito viene salvato nell'istanza `area`. I valori salvati nell'istanza `area` vengono quindi tracciati.

4. Selezionare Controllo > Prova filmato per provare il file SWF.

Nel pannello Output viene visualizzato 120.

I parametri nella funzione `getArea()` sono analoghi ai valori di una variabile locale, ovvero esistono fintanto che la funzione viene chiamata e cessano di esistere quando la funzione termina.

Nell'esempio seguente, il codice ActionScript restituisce un errore `NaN` (not a number, non un numero) se non vengono passati parametri sufficienti alla funzione `addNumbers()`.

### Per passare un numero di parametri variabile a una funzione:

1. Creare un nuovo documento Flash e salvarlo come **functionTest2.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
function addNumbers(a:Number, b:Number, c:Number):Number {
 return (a + b + c);
}
trace(addNumbers(1, 4, 6)); // 11
trace(addNumbers(1, 4)); // NaN (Not a Number), c è uguale a undefined
trace(addNumbers(1, 4, 6, 8)); // 11
```

Se non si passa un numero sufficiente di parametri alla funzione `addNumbers`, agli argomenti mancanti verrà assegnato il valore predefinito `undefined`. Se si passano troppi parametri, quelli in eccesso verranno ignorati.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Flash visualizza i seguenti valori: 11, NaN, 11.

## Restituzione di valori da funzioni

È possibile usare l'istruzione `return` affinché una funzione restituisca dei valori. L'istruzione `return` specifica il valore che verrà restituito da una funzione. L'istruzione `return` restituisce il risultato di un calcolo come valore della funzione in cui viene eseguita un'espressione.

L'istruzione `return` restituisce immediatamente il risultato al codice chiamante.

Per ulteriori informazioni, vedere `return statement` nella *Guida di riferimento di ActionScript 2.0*.

Le regole seguenti si applicano all'uso dell'istruzione `return` nelle funzioni:

- Se per una funzione si specifica un tipo restituito diverso da `Void`, è necessario includere un'istruzione `return` seguita dal valore restituito dalla funzione.
- Se si specifica un tipo restituito `Void`, non occorre includere un'istruzione `return`. Se l'istruzione viene specificata, non deve essere seguita da valori.
- Indipendentemente dal tipo restituito, un'istruzione `return` può essere utilizzata per uscire da una funzione.
- Se non si specifica un tipo `return`, l'inclusione di un'istruzione `return` è opzionale.

La funzione seguente restituisce ad esempio il quadrato del parametro `myNum` e specifica che il valore restituito deve essere di tipo `Number`:

```
function sqr(myNum:Number):Number {
 return myNum * myNum;
}
```

Alcune funzioni eseguono una serie di operazioni senza restituire un valore. Il seguente esempio restituisce il valore elaborato. Il valore viene memorizzato in una variabile che può essere utilizzata all'interno dell'applicazione.

**Per restituire un valore e memorizzarlo in una variabile:**

1. Creare un nuovo documento Flash e salvarlo come **return.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
function getArea(width:Number, height:Number):Number {
 return width * height;
}
```

La funzione `getArea` accetta due parametri: `width` e `height`.

3. Immettere il codice seguente dopo la funzione:

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

La chiamata alla funzione `getArea()` assegna i valori 10 e 12 rispettivamente ai parametri `width` e `height` e il valore restituito viene salvato nell'istanza `area`. I valori salvati nell'istanza `area` vengono quindi tracciati.

4. Selezionare Controllo > Prova filmato per provare il file SWF.

Nel pannello Output viene visualizzato 120.

I parametri nella funzione `getArea()` sono analoghi ai valori di una variabile locale, ovvero esistono fintanto che la funzione viene chiamata e cessano di esistere quando la funzione termina.

## Informazioni sulle funzioni nidificate

Una funzione può essere chiamata dall'interno di un'altra funzione. In questo modo è possibile nidificare funzioni per eseguire operazioni specifiche in Flash.

È possibile, ad esempio, nidificare funzioni in una linea temporale per eseguire operazioni specifiche su una stringa. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var myStr:String = "Il mio dolcetto è giallo.";
trace("Stringa originale: " + myStr);
function formatText():Void {
 changeString("Metti il pollo nel microonde.");
 trace("Stringa modificata: " + myStr);
}
function changeString(newtext:String):Void {
 myStr = newtext;
}
// Chiama la funzione
formatText();
```

Selezionare Controllo > Prova filmato per provare la funzione nidificata. Le due funzioni `formatText()` e `changeString()` vengono applicate alla stringa quando si chiama la funzione `formatText()`.

## Nozioni fondamentali sui metodi

I metodi sono funzioni associate a una classe che può essere una classe personalizzata o incorporata, ovvero parte del linguaggio ActionScript. Per informazioni sul confronto di metodi e funzioni, vedere “[Informazioni su funzioni e metodi](#)” a pagina 169 e “[Informazioni sui metodi e sulle funzioni](#)” a pagina 171.

Ad esempio, `sortOn()` è un metodo incorporato associato alla classe `Array` (`sortOn` è una funzione della classe predefinita `Array` incorporata in Flash).

### Per usare il metodo sortOn() in un file FLA:

1. Creare un nuovo documento Flash e salvarlo come **methods.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var userArr:Array = new Array();
userArr.push({firstname:"Giorgio", age:39});
userArr.push({firstname:"Daniele", age:43});
userArr.push({firstname:"Luca", age:2});
userArr.sortOn("firstname");
var userArrayLenth:Number = userArr.length;
var i:Number;
for (i = 0; i < userArrayLenth; i++) {
 trace(userArr[i].firstname);
}
```

Viene utilizzato il metodo `sortOn()` della classe `Array` per creare un nuovo oggetto `Array` denominato `userArr`. L'array contiene tre oggetti che, a loro volta, contengono nome ed età ed è ordinato in base al valore della proprietà `firstname` di ogni oggetto. Infine si eseguono iterazioni su ogni elemento dell'array, si visualizza il nome nel pannello Output e i nomi vengono visualizzati in ordine alfabetico.

3. Selezionare Controllo > Prova filmato per provare il file SWF.

Questo codice visualizza quanto segue nel pannello Output:

```
Daniele
Giorgio
Luca
```

Come dimostrato in “[Scrittura di funzioni con nome](#)” a pagina 175, quando si scrive il codice seguente, inserito nel fotogramma 1 della linea temporale, viene definita la funzione `eatCabbage()`.

```
function eatCabbage() {
 trace("pessimo gusto");
}
eatCabbage();
```

Tuttavia, se si scrive la funzione `eatCabbage()` in un file di classe e, ad esempio, si chiama `eatCabbage()` nel file FLA, `eatCabbage()` viene considerato un metodo.

Negli esempi seguenti viene illustrato come creare metodi in una classe:

### Per confrontare metodi e funzioni:

1. Creare un nuovo file ActionScript, quindi selezionare File > Salva con nome e salvare il file come **EatingHabits.as**.

2. Immettere il codice ActionScript seguente nella finestra Script:

```
class EatingHabits {
 public function eatCabbage():Void {
 trace("pessimo gusto");
 }
}
```

3. Salvare le modifiche in **EatingHabits.as**.

4. Creare un nuovo documento Flash, selezionare File > Salva con nome, nominarlo **methodTest.fla** e salvare il file nella stessa directory di **EatingHabits.as**.

5. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var myHabits:EatingHabits = new EatingHabits();
myHabits.eatCabbage();
```

Tramite questo codice ActionScript si chiama il metodo `eatCabbage()` della classe **EatingHabits**.

**NOTA**

Quando si utilizzano i metodi di una classe incorporata (in aggiunta alla classe personalizzata definita in precedenza in questa procedura), si utilizza un *metodo* sulla linea temporale.

6. Aggiungere il codice seguente dopo la riga di codice ActionScript riportata in precedenza:

```
function eatCarrots():Void {
 trace("ottimo gusto");
}
eatCarrots();
```

Questo codice definisce e chiama la funzione `eatCarrots()`.

7. Selezionare Controllo > Prova filmato per provare il file SWF.

## Assegnazione di nomi ai metodi

Assegnare ai metodi un nome con forma verbale, concatenando parole a lettere maiuscole e minuscole, mentre la prima lettera deve essere minuscola, ad esempio:

```
sing();
boogie();
singLoud();
danceFast();
```

Per la maggior parte dei nomi di metodi si utilizzano verbi perché i metodi eseguono operazioni su oggetti. Come con le variabili, non è possibile utilizzare caratteri speciali e i nomi dei metodi non possono iniziare con una cifra. Per ulteriori informazioni, vedere [“Convenzioni di denominazione” a pagina 793.](#)



## CAPITOLO 6

# Classi

In questo capitolo vengono introdotti l'uso e la scrittura di classi tramite ActionScript 2.0. Le classi sono la base su cui poggia ActionScript 2.0 e sono ancora più importanti in questa versione di Macromedia Flash di quanto lo fossero nelle versioni precedenti, come viene spiegato in tutto questo capitolo.

Il capitolo presenta in primo luogo la terminologia fondamentale, illustrando in che modo è associata alle classi e alla programmazione orientata agli oggetti (OOP, Object Oriented Programming). In seguito viene presentato un file di classe di esempio per dimostrare il funzionamento di ogni sezione del file e l'organizzazione della classe. Il resto del capitolo spiega come creare classi personalizzate e utilizzarle all'interno dei documenti Flash. Vengono presentati il percorso di classe di Flash e la documentazione suggerita per una classe al fine di consentire ad altri utenti che leggono o utilizzano il codice di comprendere facilmente lo scopo globale della classe.

In questa sezione sono contenuti esempi di codice utilizzabili per acquisire familiarità con la creazione di classi in ActionScript 2.0. Al termine del capitolo l'utente sarà in grado di scrivere un file di classe tipico, comprendere e riconoscere le classi Flash e leggere file di classe scritti da altri utenti.

Gli sviluppatori che non hanno mai scritto codice ActionScript 2.0 in precedenza possono consultare [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi” a pagina 75](#) e [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0” a pagina 791](#).

Per ulteriori informazioni sulle operazioni con le classi personalizzate e incorporate, consultare i seguenti argomenti:

|                                                                                                 |     |
|-------------------------------------------------------------------------------------------------|-----|
| <a href="#">Informazioni sulla programmazione orientata agli oggetti e Flash .....</a>          | 198 |
| <a href="#">Creazione di file di classi personalizzate .....</a>                                | 208 |
| <a href="#">Informazioni sulle operazioni con classi personalizzate in un'applicazione.....</a> | 211 |
| <a href="#">Esempio: creazione di classi personalizzate .....</a>                               | 237 |
| <a href="#">Esempio: uso di file di classi personalizzate in Flash .....</a>                    | 252 |
| <a href="#">Assegnazione di una classe a simboli in Flash.....</a>                              | 255 |

|                                                                    |     |
|--------------------------------------------------------------------|-----|
| Compilazione ed esportazione di classi . . . . .                   | 257 |
| Nozioni fondamentali sulle classi e l'area di validità . . . . .   | 260 |
| Informazioni sulle classi incorporate e di primo livello . . . . . | 263 |
| Informazioni sulle operazioni con le classi incorporate . . . . .  | 273 |

## Informazioni sulla programmazione orientata agli oggetti e Flash

ActionScript 2.0 è un linguaggio orientato agli oggetti. Come ActionScript, i linguaggi orientati agli oggetti si basano sul concetto di *classi* e *interfacce*. Una classe definisce tutte le proprietà che distinguono una serie di oggetti. Una classe User, ad esempio, rappresenta i diversi utenti che utilizzano l'applicazione. Vengono poi create *istanze* della classe, che, per la classe User, rappresentano ogni singolo individuo, ovvero uno dei suoi membri. Le *istanze* della classe User comprendono tutte le proprietà della classe User.

Le classi sono inoltre considerate come *tipi di dati* o *modelli* che è possibile creare per definire un nuovo tipo di oggetto. Se, ad esempio, nell'applicazione è necessario un tipo di dati Lettuce, è possibile scrivere la classe Lettuce. In questo modo si definisce l'oggetto Lettuce ed è possibile assegnare i metodi (`wash()`) e le proprietà (`leafy` o `bugs`) di Lettuce. Per definire una classe si utilizza la parola chiave `class` in un file di script esterno, creabile nello strumento di creazione di Flash selezionando File > Nuovo e quindi File ActionScript.

Con Flash Player 8, disponibile sia nella versione base di Flash 8 che in Flash Professional 8, vengono aggiunte nuove funzionalità al linguaggio ActionScript, quali ad esempio effetti filtro, il caricamento e lo scaricamento di file e l'API External. Come in precedenza, ActionScript 2.0 comprende diversi concetti e diverse parole chiave (quali `class`, `interface` e `package`) potenti e familiari per chi già utilizza la programmazione orientata agli oggetti di altri linguaggi di programmazione come Java. Il linguaggio di programmazione consente di creare strutture di programma riutilizzabili, scalabili, potenti e aggiornabili. Fornisce inoltre agli utenti assistenza nella creazione del codice e utili informazioni di debug, riducendo in tal modo il tempo di sviluppo. ActionScript 2.0 può essere utilizzato per creare oggetti e stabilire l'ereditarietà, nonché per creare classi personalizzate ed estendere le classi incorporate e di primo livello di Flash. In questo capitolo viene illustrato come creare classi e utilizzare classi personalizzate.

Flash Professional 8 e la versione base di Flash 8 e comprendono circa 65 classi incorporate e di primo livello che forniscono diversi tipi di elementi, dai tipi di dati di base (o primitivi) quali Array, Boolean, Date e così via, agli errori e agli eventi personalizzati e garantiscono modi diversi per caricare contenuto esterno (XML, immagini, dati binari originari e così via). È inoltre possibile scrivere classi personalizzate e integrarle nei documenti Flash oppure estendere le classi di primo livello e aggiungere funzionalità personalizzate o modificare le funzionalità esistenti. Ad esempio, “[Informazioni sui membri di classe](#)” a pagina 224 in questo capitolo illustra come creare una classe Person personalizzata che contenga proprietà personalizzate per il nome e l'età di una persona. La classe personalizzata può essere considerata un nuovo tipo di dati nei documenti ed è possibile creare una nuova istanza della classe tramite l'operatore new.

Per ulteriori informazioni sulle operazioni con OOP, consultare i seguenti argomenti:

- “[I vantaggi derivanti dall'utilizzo delle classi](#)” a pagina 199
- “[Informazioni sui pacchetti](#)” a pagina 200
- “[Informazioni su valori e tipi di dati](#)” a pagina 203
- “[Nozioni fondamentali sulla programmazione orientata agli oggetti.](#)” a pagina 204

## I vantaggi derivanti dall'utilizzo delle classi

Nella programmazione orientata agli oggetti una classe definisce una categoria di oggetto. Una classe descrive le proprietà (dati) e i metodi (comportamento) di un oggetto in modo molto simile a come un progetto di architettura descrive le caratteristiche di un edificio. Se si scrive una classe personalizzata in un file ActionScript (AS) esterno, è possibile importarla nell'applicazione al momento della compilazione del file FLA.

Le classi possono essere molto utili per la creazione di applicazioni Flash di grandi dimensioni perché consentono di organizzare gran parte della complessità dell'applicazione in file di classe esterni. Spostando buona parte della logica in una classe personalizzata, non solo è possibile semplificare il riutilizzo del codice, ma anche nascondere alcuni metodi e alcune proprietà ad altre porzioni del codice ActionScript per impedire agli utenti l'accesso a informazioni riservate o la modifica di dati che devono rimanere invariati.

È possibile inoltre estendere le classi esistenti modificando le funzionalità esistenti o aggiungendone di nuove. Se, ad esempio, si devono creare tre classi molto simili, è possibile scrivere una classe base e quindi due altre classi che la estendano e aggiungano metodi e proprietà, anziché creare tre file di classe che duplichino lo stesso codice e la stessa logica.

Un ulteriore vantaggio legato all'uso delle classi è rappresentato dalla riutilizzabilità del codice. Se, ad esempio, si crea una classe personalizzata per generare una barra di avanzamento tramite l'interfaccia API (Application Programming Interface) di disegno, è possibile salvare la classe della barra di avanzamento nel percorso di classe e riutilizzare lo stesso codice in tutti i documenti Flash importando la classe personalizzata. Per ulteriori informazioni sull'impostazione del percorso di classe di, vedere “[Informazioni sull'importazione di file di classe](#)” a pagina 212 e “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214.

## Informazioni sui pacchetti

Quando si creano le classi, occorre organizzare i file di classe ActionScript in *pacchetti*. Un pacchetto è una directory che contiene uno o più file di classe e che si trova in una directory prestabilita di un percorso di classe; (vedere “[Informazioni sull'importazione di file di classe](#)” a pagina 212 e “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214). Un pacchetto, a sua volta, contiene altri pacchetti denominati *sottopacchetti*, ciascuno dei quali include i rispettivi file di classe.

Analogamente a quanto avviene con le variabili, i nomi di pacchetti devono essere identificatori, ovvero il primo carattere può essere una lettera, un carattere di sottolineatura (\_) o un simbolo del dollaro (\$) e i caratteri seguenti possono essere lettere, numeri, caratteri di sottolineatura o simboli di dollaro. Per l'assegnazione di nomi ai pacchetti alcune convenzioni sono più utilizzate di altre, in base alle quali, ad esempio, è sconsigliabile evitare l'utilizzo dei caratteri di sottolineatura o dei simboli di dollaro. Per ulteriori informazioni sull'assegnazione di nomi ai pacchetti, consultare il “[Assegnazione di nomi ai pacchetti](#)” a pagina 802.

Generalmente, i pacchetti vengono utilizzati per organizzare classi correlate tra loro. Si supponga, ad esempio, di avere tre classi correlate, Square, Circle e Triangle, definite rispettivamente in Square.as, Circle.as e Triangle.as, e che i file ActionScript siano stati salvati in una directory specificata all'interno del percorso di classe, come nell'esempio seguente:

```
// In Square.as:
class Square {}

// In Circle.as:
class Circle {}

// In Triangle.as:
class Triangle {}
```

Poiché questi tre file di classe sono correlati, si potrebbe inserirli in un pacchetto (directory) denominato Shapes. In tal caso, il nome completo della classe contiene il percorso del pacchetto, oltre al nome semplice della classe. Per i percorsi dei pacchetti viene adottata la sintassi del punto (.), dove ciascun punto indica una sottodirectory.

Ad esempio, se si colloca ogni file ActionScript che definisce una forma nella directory Shapes, potrebbe essere necessario modificare il nome di ogni file di classe in modo da riflettere la nuova posizione, nel modo seguente:

```
// In Shapes/Square.as:
class Shapes.Square {}

// In Shapes/Circle.as:
class Shapes.Circle {}

// In Shapes/Triangle.as:
class Shapes.Triangle {}
```

Per indicare una classe che risiede in una directory pacchetto è possibile specificarne il nome completo della classe o importare il pacchetto utilizzando l'istruzione `import`. Per ulteriori informazioni, vedere “[Operazioni con i pacchetti](#)” a pagina 202.

## Confronto tra classi e pacchetti

Nella programmazione orientata agli oggetti una classe definisce una categoria di oggetto, Le classi sono in pratica tipi di dati che è possibile creare per definire un nuovo tipo di oggetto nell'applicazione. Una classe descrive le *proprietà* (dati) e i *comportamenti* (metodi) di un oggetto in modo molto simile a come un progetto di architettura descrive le caratteristiche di un edificio. Le proprietà (variabili definite all'interno di una classe) e i metodi di una classe sono detti *membri* della classe. Per utilizzare le proprietà e i metodi definiti da una classe, è necessario creare prima un'istanza di tale classe (a eccezione delle classi che non dispongono di membri statici; vedere “[Informazioni sui membri delle classi \(statici\)](#)” a pagina 275, ad esempio la classe di primo livello Math e “[Metodi e proprietà statici](#)” a pagina 222). La relazione tra un'istanza e la relativa classe è simile a quella che intercorre tra un edificio e il relativo progetto.

I pacchetti in Flash sono directory che contengono uno o più file di classe e che risiedono in un percorso definito. File di classi personalizzate correlati possono essere inseriti all'interno di una stessa directory. È possibile, ad esempio, organizzare tre classi correlate denominate SteelWidget, PlasticWidget e WoodWidget, definite in SteelWidget.as, PlasticWidget.as e WoodWidget.as, nel pacchetto Widget. Per ulteriori informazioni sui pacchetti, vedere “[Operazioni con i pacchetti](#)” a pagina 202 e “[Creazione di file di classe e inserimento in pacchetti](#)” a pagina 240.

## Operazioni con i pacchetti

I pacchetti sono directory che contengono uno o più file di classe e che risiedono nella directory del percorso di classe definita. Ad esempio, il pacchetto flash.filters è una directory sul disco rigido che contiene numerosi file di classe per ogni tipo di filtro (quale BevelFilter, BlurFilter, DropShadowFilter e così via) in Flash 8.



Per utilizzare l'istruzione `import`, è necessario specificare ActionScript 2.0 e Flash Player 6 o versione superiore nella scheda Flash della finestra di dialogo Impostazioni pubblicazione del file FLA.

L'istruzione `import` consente di accedere alle classi senza doverne specificare il nome completo. Ad esempio, se si desidera usare la classe `BlurFilter` in uno script, è necessario farvi riferimento con il nome completo (`flash.filters.BlurFilter`) o importarla; se la si importa, è possibile farvi riferimento con il nome della classe (`BlurFilter`). Il seguente codice ActionScript dimostra le differenze tra l'utilizzo dell'istruzione `import` e l'utilizzo dei nomi completi di classe.

Se non si importa la classe `BlurFilter`, per poter utilizzare il filtro nel codice è necessario utilizzare il nome completo della classe (nome pacchetto seguito da nome della classe):

```
// senza importazione
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10,
3);
```

Lo stesso codice, scritto con un'istruzione `import`, permette di accedere a `BlurFilter` usando soltanto il nome della classe invece del nome completo. In questo modo è possibile risparmiare tempo di scrittura e ridurre le possibilità di errori:

```
// con importazione
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Se si stanno importando più classi con un pacchetto (ad esempio, `BlurFilter`, `DropShadowFilter` e `GlowFilter`) è possibile utilizzare uno dei due metodi di importazione di ogni classe. Il primo metodo di importazione di più classi consiste nell'importare ogni classe usando un'istruzione `import` separata, come nel frammento di codice seguente:

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

L'utilizzo di singole istruzioni `import` per ogni classe in un pacchetto può richiedere un certo dispendio di tempo e può introdurre errori di battitura. Il secondo metodo di importazione di classi in un pacchetto consiste nell'utilizzare un'importazione con caratteri jolly che importa tutte le classi in un determinato livello di un pacchetto. Nel codice ActionScript seguente è contenuto un esempio dell'uso di un'importazione con caratteri jolly:

```
import flash.filters.*; // importa ogni classe nel pacchetto flash.filters
```

L'istruzione `import` si applica solo allo script corrente (fotogramma o oggetto) in cui viene chiamata. Ad esempio, se nel fotogramma 1 di un documento Flash si importano tutte le classi presenti nel pacchetto `macr.util`, su tale fotogramma, è possibile fare riferimento alle classi del pacchetto con i nomi della classe invece dei nomi completi. Se si desiderasse utilizzare il nome della classe in un altro script di fotogrammi, tuttavia, sarebbe necessario fare riferimento alle classi in tale pacchetto con i nomi completi o aggiungere un'istruzione `import` all'altro fotogramma che importa le classi nel pacchetto.

Quando si utilizzano le istruzioni `import`, è anche importante notare che le classi sono importate soltanto per il livello specificato. Ad esempio, se fossero state importate tutte le classi nel pacchetto `mx.transitions`, verrebbero importate soltanto le classi nella directory `/transitions/`, non tutte le classi nelle sottodirectory (come le classi nel pacchetto `mx.transitions.easing`).

**SUGGERIMENTO**

Se si importa una classe ma non la si utilizza nello script, la classe non viene esportata nel file SWF. Questo significa che è possibile importare pacchetti di grandi dimensioni senza doversi preoccupare delle dimensioni del file SWF; il codice byte associato a una classe viene incluso in un file SWF solo se tale classe viene effettivamente utilizzata.

## Informazioni su valori e tipi di dati

I dati, i valori e i tipi sono importanti quando si inizia a scrivere classi e a utilizzarle. Sono state fornite informazioni sui dati e i tipi nel [Capitolo 10, “Dati e tipi di dati” a pagina 333](#). Quando si lavora con le classi, tenere a mente che i tipi di dati descrivono il tipo di informazioni che può essere contenuto in una variabile o in un elemento ActionScript, ad esempio Boolean, Number e String. Per ulteriori informazioni, vedere [“Informazioni sui tipi di dati” a pagina 334](#).

Le espressioni sono associate a valori, mentre i valori e le proprietà sono associati a *tipi*. I valori impostabili e ottenibili da una proprietà nella classe devono essere compatibili con la proprietà. Per compatibilità dei tipi si intende che il tipo di un valore è compatibile con il tipo in uso, come nell'esempio seguente:

```
var myNum:Number = 10;
```

Per ulteriori informazioni sulla tipizzazione forte dei dati, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

## Nozioni fondamentali sulla programmazione orientata agli oggetti.

Prima di iniziare a scrivere codice ActionScript, nelle sezioni seguenti viene presentata la terminologia utilizzata in questo capitolo. Questa breve introduzione ai principi relativi allo sviluppo di programmi orientati agli oggetti aiuta a seguire gli esempi e le sezioni del capitolo e il resto del manuale. I principi sono descritti in modo più dettagliato nel seguito del capitolo, con informazioni sulla relativa implementazione in Flash 8.

Per dimostrare i concetti della programmazione orientata agli oggetti, le sezioni seguenti utilizzano un'analogia con un gatto.

### Oggetti

Anche un oggetto del mondo reale, ad esempio un gatto, possiede delle *proprietà* (o stati), quali nome, età e colore, e presenta delle caratteristiche comportamentali, ad esempio dormire, mangiare e fare le fusa. Analogamente, nel mondo della programmazione orientata agli oggetti, gli oggetti presentano proprietà e comportamenti. Utilizzando le tecniche orientate agli oggetti, è possibile modellare un oggetto del mondo reale (come un gatto) oppure un oggetto più astratto (ad esempio un processo chimico).

**NOTA**

Il termine *comportamenti* viene utilizzato in modo generico in questo capitolo e non si riferisce al pannello Comportamenti dell'ambiente di creazione di Flash.

Per ulteriori informazioni sugli oggetti, vedere “[Tipo di dati oggetto](#)” a pagina 341.

## Istanze e membri di classe

Procedendo con l'analogia con un gatto nel mondo reale, è possibile considerare che esistono gatti di colore, età e nomi differenti che allo stesso tempo presentano modi diversi di mangiare o di fare le fusa. Tuttavia, indipendentemente dalle differenze tra i singoli animali, tutti i gatti appartengono alla stessa categoria che, in termini di programmazione orientata agli oggetti, è detta classe. Appartengono cioè alla classe Gatto. Nella terminologia orientata agli oggetti, ogni singolo gatto viene considerato un'*istanza* della classe Gatto.

Nella programmazione orientata agli oggetti una classe definisce un modello per un tipo di oggetto. Tutte le caratteristiche e i comportamenti che appartengono a una classe sono detti *membri* di tale classe. In particolare, le caratteristiche (nell'esempio del gatto, il nome, l'età e il colore) sono dette *proprietà* della classe e sono rappresentate da variabili, mentre i comportamenti (mangiare, dormire) sono detti *metodi* della classe e sono rappresentati da funzioni.

Per ulteriori informazioni sulle istanze e i membri di classe, vedere “[Informazioni sui membri di classe](#)” a pagina 224 e “[Uso dei membri di classe](#)” a pagina 228.

## Ereditarietà

Uno dei vantaggi principali della programmazione orientata agli oggetti è rappresentato dalla creazione di *sottoclassi* (tramite *estensione* di una classe); una sottoclasse eredita tutte le proprietà e i metodi della rispettiva classe. La sottoclasse definisce generalmente ulteriori metodi e proprietà o sostituisce metodi o proprietà definiti nella superclasse. Le sottoclassi possono inoltre sostituire i metodi definiti in una superclasse, ovvero fornire definizioni proprie per tali metodi.

Uno dei vantaggi principali derivanti dall'uso di una struttura superclasse/sottoclasse è la possibilità di riutilizzare in modo semplice il codice simile tra diverse classi. È possibile, ad esempio, creare una superclasse denominata Animale che contiene caratteristiche e comportamenti comuni di tutti gli animali, quindi creare diverse sottoclassi che ereditano dalla superclasse Animale e aggiungono caratteristiche e comportamenti specifici di un determinato tipo di animale.

È possibile creare una classe Gatto che erediti da un'altra classe. È possibile, ad esempio, creare una classe Mammifero che definisce alcune proprietà e alcuni comportamenti comuni a tutti i mammiferi e, successivamente, creare una sottoclasse Gatto che estenda la classe Mammifero. Un'ulteriore sottoclasse, ad esempio la classe Siamese, potrebbe a sua volta estendere (creando una *sottoclasse*) la classe Gatto, e così via.

La creazione di sottoclassi consente di riutilizzare il codice. Invece di ricreare tutti i codici comuni a entrambe le classi, è possibile estendere semplicemente la classe esistente.

**SUGGERIMENTO**

In un'applicazione complessa, la definizione della struttura gerarchica delle classi rappresenta una parte importante del processo di progettazione e deve essere eseguita prima di iniziare la programmazione vera e propria.

Per ulteriori informazioni sull'ereditarietà e le sottoclassi, consultare il [Capitolo 7, “Ereditarietà” a pagina 279](#).

## Interfacce

Le *interfacce* nella programmazione orientata agli oggetti possono essere descritte come modelli di definizioni di classe; le classi che implementano le interfacce sono necessarie per implementare quel modello di metodo. Usando l'analogia del gatto, un'interfaccia è simile al progetto di un gatto: Il progetto evidenzia quali parti sono necessarie, ma non necessariamente come vengono assemblate le parti o qual è il funzionamento delle parti.

È possibile utilizzare interfacce per migliorare la struttura e facilitare la manutenzione delle applicazioni. Poiché ActionScript 2.0 supporta l'estensione da una sola superclasse, le interfacce possono essere utilizzate come forma limitata di ereditarietà multipla.

Un'interfaccia può anche essere considerata un "contratto di programmazione", utilizzabile per creare relazioni tra classi altrimenti non correlate. Ad esempio, si consideri una situazione di lavoro con un team di programmatore, ciascuno dei quali si occupa di una sezione (classe) diversa della stessa applicazione. In fase di progettazione dell'applicazione, viene stabilito un set di metodi che le classi utilizzeranno per comunicare. Viene quindi creata un'interfaccia che dichiara questi metodi, i relativi parametri e tipi restituiti. Qualsiasi classe che implementa questa interfaccia deve fornire le definizioni per tali metodi; in caso contrario si verificherà un errore del compilatore.

Per ulteriori informazioni sull'ereditarietà, vedere [Capitolo 7, “Ereditarietà” a pagina 279](#). Per ulteriori informazioni sulle interfacce, consultare il [Capitolo 8, “Interfacce” a pagina 293](#).

## Incapsulamento

Nella progettazione orientata agli oggetti, gli oggetti sono considerati scatole nere che contengono o *incapsulano* funzionalità. Un programmatore dovrebbe essere in grado di interagire con un oggetto conoscendone solo le proprietà, i metodi e gli eventi (l'interfaccia di programmazione), ma senza conoscerne i dettagli di implementazione. Questo approccio consente di programmare a un livello di astrazione superiore e garantisce una struttura organizzativa per la creazione di sistemi complessi.

Per garantire l'incapsulamento, ActionScript 2.0 comprende, ad esempio, il controllo dell'accesso dei membri, affinché i dettagli dell'implementazione possano essere resi privati e invisibili al codice esterno a un oggetto che deve interagire con l'interfaccia di programmazione dell'oggetto, anziché con i relativi dati di implementazione (che possono essere nascosti nei metodi e nelle proprietà privati). Questo approccio garantisce alcuni vantaggi importanti: consente ad esempio al creatore dell'oggetto di modificare l'implementazione dell'oggetto senza dover apportare modifiche al codice esterno all'oggetto, a condizione che l'interfaccia di programmazione non cambi.

Per ulteriori informazioni sull'incapsulamento, vedere “[Informazioni sull'incapsulamento](#)” a pagina 236.

## Polimorfismo

La programmazione orientata agli oggetti permette di esprimere differenze tra le singole classi con l'ausilio di una tecnica detta *polimorfismo*, grazie alla quale le classi possono sostituire i metodi delle relative superclassi e definire implementazioni specializzate di tali metodi. In Flash, le sottoclassi possono definire implementazioni specializzate di metodi ereditati dalla propria superclasse, ma non possono acceder all'implementazione della superclasse come in altri linguaggi di programmazione.

È possibile, ad esempio, creare una classe Mammifero che disponga dei metodi `giocare()` e `dormire()`, quindi creare sottoclassi Gatto, Scimmia e Cane che estendano la classe Mammifero. Le sottoclassi sostituiscono il metodo `giocare()` della classe Mammifero per riflettere le abitudini di ogni animale. Scimmia implementa il metodo `giocare()` per passare da un albero a un altro, Gatto implementa il metodo `giocare()` per far rimbalzare un gomitolo di lana, mentre Cane implementa il metodo `giocare()` per rincorrere un palla. Dato che la funzionalità `dormire()` è simile per tutti gli animali, si utilizza l'implementazione della superclasse.

Per ulteriori informazioni sul polimorfismo, consultare il Capitolo 7, “Ereditarietà” a pagina 279 e vedere “[Uso del polimorfismo in un'applicazione](#)” a pagina 287.

# Creazione di file di classi personalizzate

Nell'esempio seguente vengono esaminate le parti di un file di classe. Viene illustrato come scrivere una classe e come modificarla per poterla utilizzare in Flash in modi diversi. Vengono presentate le parti di una classe e viene indicato come importarle. Vengono inoltre fornite informazioni correlate sull'uso di file di classi personalizzate in Flash.

In primo luogo viene presentata una classe molto semplice. Nell'esempio seguente è illustrata l'organizzazione di una classe semplice denominata UserClass.

Per definire una classe si utilizza la parola chiave `class` in un file di script esterno (non in uno script in fase di creazione nel pannello Azioni). La struttura di una classe è valida anche per i file di interfaccia ed è illustrata di seguito. Successivamente si procederà alla creazione di una classe.

- Iniziare il file di classe con commenti di documentazione, tra cui una descrizione generale del codice e informazioni sull'autore e sulla versione.
- Aggiungere le istruzioni `import` (se necessarie).
- Scrivere un'istruzione relativa al pacchetto o una dichiarazione di classe o di interfaccia, come segue:

```
class UserClass {...}
```
- Includere gli eventuali commenti relativi all'implementazione della classe o dell'interfaccia, aggiungendo informazioni relative all'intera classe o all'intera interfaccia.
- Aggiungere tutte le variabili statiche. Scrivere prima le variabili di classe statiche, quindi le variabili di classe private.
- Aggiungere le variabili di istanza. Scrivere prima le variabili dei membri pubblici, quindi quelle dei membri privati.
- Aggiungere l'istruzione relativa alla funzione di costruzione, come indicato nell'esempio seguente:

```
public function UserClass(username:String, password:String) {...}
```
- Scrivere i metodi, raggruppandoli per funzionalità e non per accessibilità o area di validità. Questo tipo di organizzazione dei metodi consente di migliorare la leggibilità e la chiarezza del codice.
- Scrivere i metodi getter/setter.

L'esempio seguente illustra una classe ActionScript semplice denominata User.

**Per creare file di classe:**

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Selezionare File > Salva con nome e assegnare al file il nome **User.as**.
3. Immettere il codice ActionScript seguente nella finestra Script:

```
/**
 * Classe User
 * autore: John Doe
 * versione: 0.8
 * data modifica: 08/21/2005
 * copyright: Macromedia, Inc.

 * Questo codice definisce una classe User personalizzata per creare
 * nuovi utenti e specificare le relative informazioni di login.
 */

class User {
 // Variabili di istanza private
 private var __username:String;
 private var __password:String;

 // Istruzione relativa alla funzione di costruzione
 public function User(p_username:String, p_password:String) {
 this.__username = p_username;
 this.__password = p_password;
 }

 public function get username():String {
 return this.__username;
 }
 public function set username(value:String):Void {
 this.__username = value;
 }

 public function get password():String {
 return this.__password;
 }
 public function set password(value:String):Void {
 this.__password = value;
 }
}
```

4. Salvare le modifiche apportate al file di classe.

Il frammento di codice precedente inizia con un *commento di documentazione* standard che specifica il nome della classe, l'autore, la versione, la data di modifica, le informazioni di copyright e una breve descrizione dello scopo della classe.

L'istruzione della funzione di costruzione della classe User accetta due parametri: `p_username` e `p_password` che vengono copiati nelle variabili di istanza private della classe `__username` e `__password`. Il resto del codice della classe definisce le proprietà `getter` e `setter` per le variabili di istanza private. Per creare una proprietà di sola lettura, definire una funzione `getter` senza una funzione `setter`. Se, ad esempio, si desidera che un nome utente non venga modificato in seguito alla definizione, eliminare la funzione `setter` `username` dal file di classe User.

5. Selezionare File > Nuovo, quindi Documento Flash.
6. Selezionare File > Salva con nome e assegnare al file il nome **user\_test.fla**. Salvare il file nella stessa directory di User.as.
7. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
import User;
var user1:User = new User("un1", "pw1");
trace("Prima:");
trace("\t username = " + user1.username); // un1
trace("\t password = " + user1.password); // pw1
user1.username = "lnu";
user1.password = "lwp";
trace("Dopo:");
trace("\t username = " + user1.username); // lnu
trace("\t password = " + user1.password); // lwp
```

Il codice ActionScript nel documento Flash è molto semplice, data la semplicità della classe User creata in precedenza. La prima riga di codice importa la classe User personalizzata nel documento Flash per poterla utilizzare come tipo di dati personalizzato.

Viene definita un'istanza della classe User che viene assegnata alla variabile `user1`. All'oggetto `user1` di User viene assegnato un valore e vengono definiti uno `username` `un1` e una `password` `pw1`. Le due istruzioni `trace` seguenti visualizzano il valore corrente di `user1.username` e `user1.password` tramite le funzioni `getter` della classe User che restituiscono stringhe. Le due righe successive utilizzano le funzioni `setter` della classe User per impostare nuovi valori per le variabili `username` e `password`. Infine, i valori di `username` e `password` vengono tracciati e visualizzati nel pannello Output. Le istruzioni `trace` visualizzano i valori modificati impostati tramite le funzioni `setter`.

8. Salvare il file FLA e selezionare Controllo > Prova filmato per provare i file.

Nel pannello Output vengono visualizzati i risultati delle istruzioni `trace`. Negli esempi che seguono, i file verranno utilizzati in un'applicazione.

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `XmlMenu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, `xmlmenu.fla`, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML\_Menu*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples\ActionScript/XML\_Menu*.

## Informazioni sulle operazioni con classi personalizzate in un'applicazione

In “[Creazione di file di classi personalizzate](#)” a pagina 208 è stato creato un file di classe personalizzata. Nelle sezioni seguenti il file verrà utilizzato in un'applicazione. Il flusso di lavoro necessario per la creazione di classi comprende almeno i punti seguenti:

1. Definizione di una classe in un file di classe ActionScript esterno. Per informazioni sulla definizione e la scrittura di un file di classe, vedere “[Creazione di file di classi personalizzate](#)” a pagina 208.
2. Salvataggio del file di classe nella directory specificata per il percorso della classe (o nel percorso in cui Flash cerca le classi) oppure nella stessa directory del file FLA dell'applicazione. Per ulteriori informazioni sull'impostazione del percorso di classe, vedere “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214. Per un confronto e ulteriori informazioni sull'importazione di file di classe, vedere “[Informazioni sull'importazione di file di classe](#)” a pagina 212.
3. Creazione di un'istanza della classe in un altro script, ossia un documento FLA o un file di script esterno, oppure tramite creazione di una sottoclasse basata sulla classe originale. Per ulteriori informazioni sulla creazione di un'istanza di una classe, vedere “[Creazione di istanze di classi in un esempio](#)” a pagina 254.

Nelle seguenti sezioni del capitolo sono contenuti esempi di codice utilizzabili per acquisire familiarità con la creazione di classi in ActionScript 2.0. Gli sviluppatori che non hanno mai scritto codice ActionScript 2.0 in precedenza possono consultare il [Capitolo 10, “Dati e tipi di dati”](#) a pagina 333 e il [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi”](#) a pagina 75.

Per ulteriori informazioni sull'utilizzo delle classi, consultare i seguenti argomenti:

- “[Informazioni sull'importazione di file di classe](#)” a pagina 212
- “[Uso di un file di classe in Flash](#)” a pagina 217
- “[Uso di metodi e proprietà da un file di classe](#)” a pagina 218
- “[Informazioni sui membri di classe](#)” a pagina 224
- “[Informazioni sui metodi getter e setter](#)” a pagina 229
- “[Procedura del compilatore per la risoluzione dei riferimenti alle classi](#)” a pagina 217
- “[Informazioni sulle classi dinamiche](#)” a pagina 233
- “[Informazioni sull'incapsulamento](#)” a pagina 236
- “[Informazioni sull'uso della parola chiave this nelle classi](#)” a pagina 237

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `Xm|Menu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, `xmlmenu.fla`, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML\_Menu*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples\ActionScript/XML\_Menu*.

## Informazioni sull'importazione di file di classe

Per utilizzare una classe o un'interfaccia già definita, Flash deve individuare i file ActionScript esterni che contengono la definizione della classe o dell'interfaccia per importare il file.

L'elenco delle directory in cui Flash esegue la ricerca delle definizioni di classe, interfaccia, funzione e variabile è detto *percorso di classe*. Flash dispone di due impostazioni del percorso di classe, un percorso di classe globale e un percorso di classe a livello di documento:

- Per **percorso di classe globale** si intende un percorso di classe condiviso da tutti i documenti Flash. Lo si imposta nella finestra di dialogo Preferenze (Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh), fare clic su ActionScript nell'elenco Categorie e quindi fare clic su Impostazioni di ActionScript 2.0).

- Per **percorso di classe a livello di documento** si intende un percorso di classe definito in modo specifico per un solo documento Flash. Lo si imposta nella finestra di dialogo Impostazioni pubblicazione (File > Impostazioni pubblicazione, selezionare la scheda Flash e quindi fare clic sul pulsante Impostazioni).

Quando si importano file di classe, sono valide le regole seguenti:

- Le istruzioni `import` possono essere inserite nelle posizioni seguenti:
  - in un punto qualsiasi prima della definizione della classe nei file di classe
  - in un punto qualsiasi negli script di fotogramma o oggetto
  - In un punto qualsiasi nei file di ActionScript che vengono inclusi in un'applicazione (usando l'istruzione `#include`).
- Per importare definizioni di pacchetto singole, utilizzare la sintassi seguente:  
`import flash.display.BitmapData;`
- Per importare diversi pacchetti, è possibile utilizzare la sintassi dei caratteri jolly:  
`import flash.display.*;`

È anche possibile includere un codice di ActionScript in un file di documento Flash (FLA) usando un'istruzione `include`. Per l'istruzione `include` sono valide le regole seguenti:

- Le istruzioni `include` corrispondono praticamente a un'operazione di copia e incolla di contenuto all'interno del file ActionScript incluso.
- Le istruzioni `include` all'interno di un file di classe di ActionScript sono relative alla sottodirectory che contiene il file.
- Un'istruzione `include` in un file FLA può importare solo codice valido all'interno di file FLA. La stessa regola è valida per gli altri punti in cui vengono inserite le istruzioni `include`. Se, ad esempio, si inserisce un'istruzione `include` all'interno di una definizione di classe, nel file ActionScript incluso possono essere presenti solo definizioni di metodi e proprietà:

```
// Foo.as
class Foo {
 #include "FooDef.as"
}

// FooDef.as:
var fooProp;
function fooMethod() {}
trace("Foo"); // Istruzione non consentita in una definizione di classe.
```

Per ulteriori informazioni sull'istruzione `include`, vedere `#include` directive nella *Guida di riferimento di ActionScript 2.0*. Per ulteriori informazioni sui percorsi di classe, vedere “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214.

## Informazioni sull'impostazione e la modifica del percorso di classe

Per utilizzare una classe o un'interfaccia già definita, Flash deve individuare i file ActionScript esterni che contengono la definizione della classe o dell'interfaccia. L'elenco delle directory in cui Flash esegue la ricerca delle definizioni di classe o di interfaccia è detto *percorso di classe*.

Quando si crea un file di classe ActionScript, è necessario salvarlo in una delle directory specificate nel percorso di classe o in una relativa sottodirectory. È possibile modificare il percorso di classe in modo che includa il percorso della directory desiderata. In caso contrario, Flash non è in grado di risolvere, ovvero di localizzare, la classe o l'interfaccia specificata nello script. Le sottodirectory create all'interno della directory di un percorso di classe sono dette pacchetti e consentono di organizzare le classi. Per ulteriori informazioni sui pacchetti, vedere ["Creazione di file di classe e inserimento in pacchetti" a pagina 240](#).

Flash ha due impostazioni per il percorso di classe: un *percorso di classe globale* e uno *a livello di documento*. Per percorso di classe globale si intende un percorso di classe condiviso da tutti i documenti Flash. Per percorso di classe a livello di documento si intende un percorso di classe definito in modo specifico per un solo documento Flash.

Il percorso di classe globale è valido per i file ActionScript esterni e per i file FLA e può essere impostato nella finestra di dialogo Preferenze (Windows: Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh), selezionare ActionScript nell'elenco Categoria e quindi fare clic su Impostazioni di ActionScript 2.0). Il percorso di classe a livello di documento può essere impostato nella finestra di dialogo Impostazioni pubblicazione (File > Impostazioni pubblicazione, selezionare la scheda Flash e fare clic sul pulsante Impostazioni).

**NOTA**

Quando si fa clic sul pulsante Controlla sintassi nella parte superiore del pannello Script durante la modifica di un file ActionScript, il compilatore esegue la ricerca solo nel percorso di classe globale. I file ActionScript non sono associati ai file FLA in modalità di modifica e non dispongono di un percorso di classe proprio.

### Uso di un percorso di classe globale

Per percorso di classe globale si intende un percorso di classe condiviso da tutti i documenti Flash.

È possibile modificare il percorso di classe globale utilizzando la finestra di dialogo Preferenze. Per modificare le impostazioni del percorso di classe relativo al documento, utilizzare la finestra di dialogo Impostazioni pubblicazione per il file FLA. In entrambi i casi è possibile aggiungere percorsi di directory assoluti, ad esempio C:/my\_classes, e relativi, ad esempio, ../my\_classes o ". ". L'ordine delle directory nella finestra di dialogo riflette l'ordine della ricerca.

Per impostazione predefinita, il percorso di classe globale contiene un percorso assoluto e un percorso relativo. Il percorso assoluto è indicato con \$(LocalData)/Classes nella finestra di dialogo Preferenze e corrisponde a quanto segue:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\Classes.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/Classes.

**NOTA**

Non eliminare il percorso di classe globale assoluto perché viene utilizzato da Flash per accedere alle classi incorporate. Se il percorso è stato eliminato inavvertitamente, reinserirlo aggiungendo \$(LocalData)/Classes come nuovo percorso di classe.

La parte del percorso di classe globale corrispondente al percorso relativo è indicata da un punto singolo (.) e punta alla directory del documento corrente. I percorsi di classe relativi possono puntare a directory differenti a seconda del percorso del documento compilato o pubblicato.

Per aggiungere un percorso di classe globale o modificarne uno esistente, attenersi alle indicazioni fornite di seguito.

**Per modificare il percorso di classe globale:**

1. Selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh) per visualizzare la finestra di dialogo Preferenze.
2. Fare clic su ActionScript nella colonna sinistra, quindi sul pulsante Impostazioni ActionScript 2.0.
3. Fare clic sul pulsante Trova il percorso per cercare la directory da aggiungere.
4. Passare al percorso da aggiungere e fare clic su OK.

**Per eliminare una directory dal percorso di classe:**

1. Selezionare il percorso nell'elenco dei percorsi di classe.
2. Fare clic sul pulsante Elimina il percorso selezionato.

**NOTA**

Non eliminare il percorso di classe globale assoluto perché viene utilizzato da Flash per accedere alle classi incorporate. Se il percorso è stato eliminato inavvertitamente, è possibile reinserirlo aggiungendo \$(LocalData)/Classes come nuovo percorso di classe.

Per informazioni sull'importazione di pacchetti, vedere “[Operazioni con i pacchetti](#)” a pagina 202.

## Uso di un percorso di classe a livello di documento

Il percorso di classe a livello di documento è valido solo per i file FLA. Si imposta il percorso di classe a livello di documento nella finestra di dialogo Impostazioni pubblicazione per un determinato file FLA (File > Impostazioni pubblicazione, fare clic sulla scheda Flash e quindi su Impostazioni di ActionScript 2.0). Per impostazione predefinita, il percorso di classe relativo a un documento è vuoto. Quando si crea un file FLA e lo si salva in una directory, questa diventa la directory prestabilita del percorso di classe.

Nei seguenti casi potrebbe essere utile salvare le classi create in una directory che verrà aggiunta all'elenco delle directory del percorso di classe globale:

- Se è stato impostato un set di classi di utilità utilizzate da tutti i progetti
- Se si desidera controllare la sintassi del codice (tramite il pulsante Controlla sintassi) presente all'interno del file ActionScript esterno

La creazione di una directory evita di perdere le classi personalizzate in caso di disinstallazione e reinstallazione di Flash, in particolare se la directory del percorso di classe globale viene eliminata e sovrascritta, perché in questo caso si perderebbero tutte le classi contenute in tale directory.

È possibile, ad esempio, creare una directory come la seguente per le proprie classi personalizzate:

- Windows: Disco rigido\Documents and Settings\utente\classi personalizzate.
- Macintosh: Disco rigido\Utenti\utente\classi personalizzate.

Questo percorso viene quindi aggiunto all'elenco dei percorsi di classe globali. Vedere “[Uso di un percorso di classe globale](#)” a pagina 214.

Quando Flash tenta di risolvere i riferimenti alle classi in uno script FLA, esegue la ricerca in primo luogo nel percorso di classe a livello di documento specificato per il file FLA. Se Flash non trova la classe in questo percorso di classe o se il percorso è vuoto, la ricerca viene eseguita nel percorso di classe globale. Se la classe non viene trovata nel percorso di classe globale, si verifica un errore del compilatore.

### Per modificare il percorso di classe al livello di documento:

1. Selezionare File > Impostazioni pubblicazioni per aprire la finestra di dialogo Impostazioni pubblicazioni.
2. Fare clic sulla scheda Flash.
3. Fare clic sul pulsante Impostazioni, in corrispondenza del menu a comparsa Versione di ActionScript.

4. È possibile specificare un percorso di file manualmente oppure fare clic sul pulsante Trova il percorso per cercare la directory da aggiungere al percorso di classe.

NOTA

Per modificare una directory di un percorso di classe esistente, selezionare il percorso nell'elenco del percorso di classe, fare clic sul pulsante Trova il percorso, individuare la directory da aggiungere e fare clic su OK.

NOTA

Per eliminare una directory dal percorso di classe, selezionarla nell'elenco del percorso di classe e fare clic sul pulsante Elimina il percorso selezionato (-).

Per ulteriori informazioni sui pacchetti, vedere “[Informazioni sui pacchetti](#)” a pagina 200.

## Procedura del compilatore per la risoluzione dei riferimenti alle classi

Quando Flash tenta di risolvere i riferimenti alle classi in uno script FLA, esegue la ricerca in primo luogo nel percorso di classe a livello di documento specificato per il file FLA. Se la classe non viene trovata in questo percorso di classe o se il percorso è vuoto, la ricerca viene eseguita nel percorso di classe globale. Se la classe non viene trovata all'interno del percorso di classe globale, si verifica un errore del compilatore.

In Flash Professional, quando si fa clic sul pulsante Controlla sintassi durante la modifica di un file ActionScript, il compilatore cerca solo nel percorso di classe globale. I file ActionScript non sono associati ai file FLA in modalità di modifica e non dispongono di un percorso di classe proprio.

## Uso di un file di classe in Flash

Per creare un'istanza di una classe ActionScript, utilizzare l'operatore `new` per richiamare la funzione di costruzione della classe. Tale funzione ha sempre lo stesso nome della classe e restituisce un'istanza della classe che generalmente viene assegnata a una variabile. Se, ad esempio, si utilizza la classe User di “[Creazione di file di classi personalizzate](#)” a pagina 208, per creare un nuovo oggetto User, scrivere il codice seguente:

```
var firstUser:User = new User();
```

NOTA

In alcuni casi non occorre creare un'istanza di una classe per utilizzarne i metodi e le proprietà. Per ulteriori informazioni sui membri di classe (statici), vedere “[Informazioni sui membri delle classi \(statici\)](#)” a pagina 275 e “[Metodi e proprietà statici](#)” a pagina 222.

È possibile usare l'operatore punto (.) per accedere al valore di una proprietà in un'istanza. Specificare il nome dell'istanza a sinistra del punto e il nome della proprietà a destra. Ad esempio, nell'istruzione seguente `firstUser` è l'istanza, mentre `username` è la proprietà:  
`firstUser.username`

È anche possibile utilizzare in un documento Flash le classi incorporate o di primo livello che fanno parte del linguaggio ActionScript. Il codice seguente, ad esempio, crea un nuovo oggetto Array, quindi ne visualizza la proprietà `length`:

```
var myArray:Array = new Array("mele", "arance", "banane");
trace(myArray.length); // 3
```

Per ulteriori informazioni sulla creazione di classi personalizzate in Flash, vedere “[Esempio: uso di file di classi personalizzate in Flash](#)” a pagina 252. Per informazioni sulla funzione di costruzione, vedere “[Creazione della funzione di costruzione](#)” a pagina 243.

## Uso di metodi e proprietà da un file di classe

Nella programmazione orientata agli oggetti, i membri (proprietà o metodi) di una classe possono essere membri di istanza o membri di classe. I membri di istanza vengono creati per ogni istanza della classe e vengono definiti con il prototipo della classe se sono dichiarati nella definizione della classe. I membri di classe, invece, vengono creati una sola volta per ogni classe. (I membri di una classe sono detti anche membri statici).

I metodi sono attributi che definiscono un oggetto. Ad esempio, `length` è una proprietà di tutti gli array che specifica il numero di elementi nell'array. I metodi sono funzioni associate a una classe. Per ulteriori informazioni su funzioni e metodi, consultare il [Capitolo 5, “Funzioni e metodi”](#) a pagina 169

L'esempio seguente illustra come creare un metodo in un file di classe:

```
class Sample {
 public function myMethod():Void {
 trace("myMethod");
 }
}
```

In seguito è possibile richiamare il metodo nel documento. Per richiamare il metodo di un'istanza o accedere alla proprietà di un'istanza, fare riferimento a un'istanza della classe. Nell'esempio seguente, `picture01`, un'istanza della classe personalizzata Picture (disponibile nell'esercizio seguente), richiama il metodo `showInfo()`:

```
var img1:Picture = new Picture("http://www.helpexamples.com/flash/images/
 image1.jpg");
// Richiama il metodo showInfo()
img1.showInfo();
```

L'esempio seguente dimostra come scrivere una classe Picture personalizzata per memorizzare diverse informazioni su una foto.

**Per utilizzare le classi Picture e PictureClass in un file FLA:**

1. Selezionare File > Nuovo, quindi selezionare File ActionScript. Salvare il documento come **Picture.as** e quindi fare clic su OK.

Scrivere la classe Picture personalizzata nel documento.

2. Immettere il codice ActionScript seguente nella finestra Script:

```
/**
 * Classe Picture
 * autore: John Doe
 * versione: 0.53
 * data modifica: 6/24/2005
 * copyright: Macromedia, Inc.

 * La classe Picture viene utilizzata come contenitore per un'immagine e
 * il relativo URL.
 */

class Picture {
 private var __infoObj:Object;

 public function Picture(src:String) {
 this.__infoObj = new Object();
 this.__infoObj.src = src;
 }

 public function showInfo():Void {
 trace(this.toString());
 }
 private function toString():String {
 return "[Picture src=" + this.__infoObj.src + "]";
 }

 public function get src():String {
 return this.__infoObj.src;
 }
 public function set src(value:String):Void {
 this.__infoObj.src = value;
 }
}
```

3. Salvare il file ActionScript.

4. Selezionare File > Nuovo, quindi Documento Flash per creare un nuovo file FLA. Salvarlo come **picture\_test.fla** nella stessa directory in cui è stato salvato il file della classe Picture.

**5.** Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
var picture1:Picture = new Picture("http://www.helpexamples.com/flash/
 images/image1.jpg");
picture1.showInfo();
this.createEmptyMovieClip("img_mc", 9);
img_mc.loadMovie(picture1.src);
```

**6.** Salvare il documento Flash.

**7.** Selezionare Controllo > Prova filmato per provare il documento.

Il testo seguente viene visualizzato nel pannello Output:

```
[Picture src=http://www.helpexamples.com/flash/images/image1.jpg]
```

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `XmlMenu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, `xmlmenu.fla`, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML\_Menu*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples\ActionScript/XML\_Menu*.

## Informazioni su metodi e proprietà (membri) pubblici, privati e statici

Nei file di classe ActionScript che vengono inseriti in un file di script esterno è possibile creare quattro tipi di metodi e proprietà: proprietà e metodi pubblici, proprietà e metodi privati, proprietà e metodi statici pubblici e proprietà e metodi statici privati. I metodi e le proprietà definiscono in che modo Flash può accedere alle variabili e consentono di specificare le parti di codice che possono accedere a determinati metodi o proprietà.

Per la creazione di applicazioni basate su classi, indipendentemente dalle dimensioni dell'applicazione, è importante soprattutto valutare attentamente se una proprietà o un metodo deve essere privato o pubblico per garantire la massima protezione del codice. Se, ad esempio, si crea una classe User, è probabile che si desideri che gli utenti che utilizzano la classe non siano in grado di modificare l'ID utente. Impostando la proprietà della classe (detta a volte *membro di istanza*) come `privata`, è possibile limitare l'accesso alla proprietà al codice all'interno della classe o delle relative sottoclassi. In tal modo gli utenti non possono modificare direttamente la proprietà.

## Proprietà e metodi pubblici

La parola chiave `public` specifica che una variabile o una funzione è disponibile per qualsiasi chiamante. Poiché le variabili e le funzioni sono pubbliche per impostazione predefinita, viene utilizzata la parola chiave `this` principalmente per ragioni di stile e leggibilità, al fine di indicare che la variabile esiste nell'area di validità corrente. È possibile, ad esempio, utilizzare la parola chiave `this` per motivi di coerenza in un blocco di codice che contiene anche variabili private o statiche. La parola chiave `this` può essere usata sia con la parola chiave `public` sia con la parola chiave `private`.

La classe Sample seguente contiene già un metodo pubblico denominato `myMethod()`:

```
class Sample {
 private var ID:Number;
 public function myMethod():Void {
 this.ID = 15;
 trace(this.ID); // 15
 trace("myMethod");
 }
}
```

Se si desidera aggiungere una proprietà di tipo `public`, utilizzare la parola "public" anziché "private", come nell'esempio seguente:

```
class Sample {
 private var ID:Number;
 public var email:String;
 public function myMethod():Void {
 trace("myMethod");
 }
}
```

Dato che la proprietà `email` è pubblica, è possibile modificarla all'interno della classe Sample o direttamente all'interno di un file FLA.

## Proprietà e metodi privati

La parola chiave `private` specifica che una variabile o una funzione è disponibile solo per la classe che la dichiara o la definisce o per le relative sottoclassi. Per impostazione predefinita una variabile o una funzione è pubblica e disponibile per qualsiasi chiamante. Utilizzare la parola chiave `this` se si desidera limitare l'accesso a una variabile o funzione, come nell'esempio seguente:

```
class Sample {
 private var ID:Number;
 public function myMethod():Void {
 this.ID = 15;
 trace(this.ID); // 15
 trace("myMethod");
 }
}
```

Per aggiungere una proprietà privata alla classe precedente, è sufficiente utilizzare la parola chiave `private` prima della parola chiave `var`.

Se si tenta di accedere alla proprietà privata `ID` dall'esterno della classe `Sample`, si ottiene un errore di compilazione e viene visualizzato un messaggio nel pannello Output. Il messaggio indica che il membro è privato e non accessibile.

## Metodi e proprietà statici

La parola chiave `static` specifica che una variabile o una funzione viene creata solo una volta per ogni classe anziché in ogni oggetto basato sulla classe. È possibile accedere a un membro di classe statico senza creare un'istanza della classe. I metodi e le proprietà statici possono essere impostati nell'area di validità pubblica o privata.

Questi metodi, detti anche *membri di classe*, vengono assegnati alla classe e non a una sua istanza. Per richiamare un metodo della classe o per accedere a una sua proprietà, fare riferimento al nome della classe anziché a una sua istanza specifica, come nel codice seguente:

```
trace(Math.PI / 8); // 0,392699081698724
```

Se si immette questa riga di codice nel pannello Script del pannello Azioni, viene visualizzato il risultato della traccia nel pannello Output.

Nell'esempio della classe Sample precedente è possibile creare una variabile statica per tenere traccia del numero di istanze della classe create, come dimostrato di seguito:

```
class Sample {
 public static var count:Number = 0;
 private var ID:Number;
 public var email:String;
 public function Sample() {
 Sample.count++;
 trace("contatore aggiornato: " + Sample.count);
 }
 public function myMethod():Void {
 trace("myMethod");
 }
}
```

Ogni volta che viene creata una nuova istanza della classe Sample, il metodo della funzione di costruzione traccia il numero totale di istanze di classi Sample definite.

Alcune classi ActionScript di primo livello dispongono di membri di classe (o statici), come illustrato in precedenza in questa sezione quando è stata chiamata la proprietà `Math.PI`. I membri di classe (proprietà e metodi) vengono richiamati non sull'istanza della classe, ma sul nome della classe stessa ed è possibile accedervi nello stesso modo. Non è pertanto necessario creare un'istanza della classe per utilizzarne metodi e proprietà.

Ad esempio, la classe di primo livello `Math` è costituita solo da proprietà e metodi statici. Per chiamare uno dei suoi metodi non occorre creare un'istanza della classe. Al contrario, è sufficiente chiamare i metodi sulla classe `Math` stessa. Il codice seguente chiama il metodo `sqrt()` della classe `Math`:

```
var squareRoot:Number = Math.sqrt(4);
trace(squareRoot); // 2
```

Il codice seguente richiama il metodo `max()` della classe `Math` che determina quale tra due numeri è il maggiore:

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Per ulteriori informazioni sulla creazione di membri di classe, vedere “[Informazioni sui membri di classe](#)” a pagina 224 e “[Uso dei membri di classe](#)” a pagina 228.

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `XmlMenu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, xmlmenu.fla, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML\_Menu*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/XML\_Menu*.

## Informazioni sui membri di classe

Molti dei membri (metodi e proprietà) presentati finora in questo capitolo sono *membri di istanza*. In ogni istanza della classe è presente una copia univoca di ogni membro di istanza. La variabile membro `email` della classe `Sample`, ad esempio, dispone di un membro di istanza, perché ogni persona ha un indirizzo e-mail diverso.

Esiste tuttavia un altro tipo di membro, vale a dire il *membro di classe*. Per ogni classe viene utilizzata una sola copia di un membro di classe. Qualsiasi variabile dichiarata all'interno di una classe, ma all'esterno di una funzione, rappresenta una proprietà della classe. Nell'esempio seguente, la classe `Person` dispone di due proprietà, `age` e `username`, a cui vengono assegnati rispettivamente i tipi `Number` e `String`:

```
class Person {
 public var age:Number;
 public var username:String;
}
```

Analogamente, qualsiasi funzione dichiarata all'interno di una classe è considerata un metodo della classe. Nell'esempio della classe `Person`, è possibile creare un metodo denominato `getInfo()`:

```
class Person {
 public var age:Number;
 public var username:String;
 public function getInfo():String {
 // Definizione del metodo getInfo()
 }
}
```

Nel frammento di codice precedente, il metodo `getInfo()` della classe `Person` e le proprietà `age` e `username` sono membri di istanza pubblici. La proprietà `age` non sarebbe un membro di classe corretto perché, come già visto, ogni persona ha un'età diversa. Solo le proprietà e i metodi comuni a tutti i membri della classe possono essere membri di classe.

Se, ad esempio, ogni classe dispone di una variabile `species` che indica il nome latino della specie rappresentata dalla classe, per ogni oggetto Person la specie sarebbe *Homo sapiens*. Dato che non sarebbe efficiente memorizzare una copia univoca della stringa "Homo sapiens" per ogni istanza della classe, questo sarà un membro di classe.

I membri di classe vengono dichiarati con la parola chiave `static`. Il membro di classe `species`, ad esempio, potrebbe essere dichiarato con il codice seguente:

```
class Person {
 public static var species:String = "Homo sapiens";
 // ...
}
```

I metodi di una classe possono anche essere dichiarati come statici, come nel codice seguente:

```
public static function getSpecies():String {
 return Person.species;
}
```

I metodi statici possono accedere solo alle proprietà statiche e non alle proprietà di istanza. Il seguente codice, ad esempio, causa un errore del compilatore, perché il metodo di classe `getAge()` fa riferimento alla variabile di istanza `age`:

```
class Person {
 public var age:Number = 15;
 // ...
 public static function getAge():Number {
 return age; /* **Errore**: Impossibile accedere alle variabili di
 istanza nelle funzioni statiche. */
 }
}
```

Per risolvere questo problema, è possibile definire il metodo come metodo di un'istanza o la variabile come variabile di una classe.

Per ulteriori informazioni sui membri di classe (detti anche proprietà statiche), vedere "[Metodi e proprietà statici](#)" a pagina 222.

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `XmlMenu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, xmlmenu.fla, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in *unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML\_Menu*.
- In Macintosh: *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples/ActionScript/XML\_Menu*.

## Uso del modello di progettazione Singleton

Il modello di progettazione Singleton prevede un uso piuttosto comune dei membri di classe. Un *modello di progettazione* definisce un approccio formale alla strutturazione del codice e può rappresentare una soluzione per i problemi di programmazione che si riscontrano comunemente. Il modello di progettazione Singleton è solo uno dei diversi modelli esistenti. In base a questo modello, ogni classe deve avere una sola istanza e per accedere globalmente all'istanza viene fornito un metodo comune. Per informazioni dettagliate sul modello di progettazione Singleton, vedere [www.macromedia.com/devnet/mx/coldfusion/articles/design\\_patterns.html](http://www.macromedia.com/devnet/mx/coldfusion/articles/design_patterns.html).

In molte situazioni è necessario disporre di un oggetto specifico di un tipo specifico. Nel gioco degli scacchi, ad esempio, è disponibile una sola scacchiera, mentre in un Paese c'è una sola capitale. Sebbene sia presente un solo oggetto, le sue funzionalità dovrebbero essere incorporate in una classe, gestendo l'unica istanza dell'oggetto e accedendo alla stessa. A questo scopo è possibile fare ricorso a una variabile globale, tuttavia per molti progetti questo metodo non è consigliabile. È sicuramente meglio far sì che la classe gestisca l'unica istanza dell'oggetto in modo autonomo tramite i membri di classe. L'esempio seguente illustra un tipico utilizzo di un modello di progettazione Singleton, in cui l'istanza Singleton viene creata una sola volta.

**Per utilizzare il modello di progettazione Singleton:**

1. Selezionare File > Nuovo, quindi selezionare File ActionScript. Salvare il documento come **Singleton.as**.

2. Immettere il codice ActionScript seguente nella finestra Script:

```
/**
 * Classe Singleton
 * autore: John Doe
 * versione: 0.53
 * data modifica: 6/24/2008
 * copyright: Macromedia, Inc.
 */

class Singleton {
 private static var instance:Singleton = null;
 public function trackChanges():Void {
 trace("rilevamento delle modifiche.");
 }
 public static function getInstance():Singleton {
 if (Singleton.instance == null) {
 trace("creazione nuovo Singleton.");
 Singleton.instance = new Singleton();
 }
 return Singleton.instance;
 }
}
```

3. Salvare il documento **Singleton.as**.

4. Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA e salvarlo come **singleton\_test.fla** nella stessa directory in cui è stato salvato il file di classe Singleton.

5. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
Singleton.getInstance().trackChanges(); // rilevamento delle modifiche.

var s:Singleton = Singleton.getInstance(); // rilevamento delle modifiche.
s.trackChanges();
```

6. Salvare il documento Flash.

7. Selezionare Controllo > Prova filmato per provare il documento.

L'oggetto Singleton non viene creato fino a quando non è effettivamente necessario, ovvero fino a quando un elemento di codice non lo richiede, chiamando il metodo `getInstance()`. Questo comportamento è detto *creazione lazy* e in molte circostanze può aiutare a migliorare l'efficienza del codice.

Non utilizzare un numero troppo elevato o troppo ridotto di classi nell'applicazione in quanto si creerebbero molti file di classe progettati in modo non corretto e le prestazioni dell'applicazione o il flusso di lavoro ne risulterebbero compromessi. È sempre consigliabile cercare di utilizzare i file di classe invece di inserire altrove parti di codice (come con le linee temporali); tuttavia, evitare di creare troppe classi che hanno soltanto poche funzionalità o soltanto poche classi che gestiscono molte funzionalità. Entrambi questi scenari rivelano una progettazione non ottimale.

## Uso dei membri di classe

Un utilizzo dei membri (statici) della classe consiste nel gestire le informazioni sullo stato relative a una classe e alle relative istanze. Ad esempio, se si desidera tenere traccia del numero di istanze create da una classe specifica. Per eseguire questa operazione in modo semplice è possibile utilizzare una proprietà della classe che viene incrementata ogni volta che si crea una nuova istanza.

Nell'esempio seguente, viene creata una classe denominata Widget che definisce un contatore di istanza statico, denominato widgetCount. Ogni volta che viene creata una nuova istanza della classe, il valore di widgetCount viene incrementato di 1 e il valore corrente di widgetCount viene visualizzato nel pannello Output.

### Per creare un contatore di istanza utilizzando una variabile di classe:

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Immettere il codice seguente nella finestra Script:

```
class Widget {
 // Inizializza la variabile di classe
 public static var widgetCount:Number = 0;
 public function Widget() {
 Widget.widgetCount++;
 trace("Creazione widget n." + Widget.widgetCount);
 }
}
```

La variabile widgetCount viene dichiarata come statica e quindi si inizializza da 0 solo una volta. Ogni volta che viene chiamata l'istruzione della funzione di costruzione della classe Widget, viene aggiunto 1 a widgetCount e visualizzato il numero dell'istanza corrente in corso di creazione.

3. Salvare il file come **Widget.as**.
4. Selezionare File > Nuovo e quindi Documento Flash per creare un nuovo file FLA e salvarlo come **widget\_test.fla** nella stessa directory del file Widget.as.

- 5.** In `widget_test.fla`, immettere il codice seguente nel fotogramma 1 della linea temporale:

```
//Prima di creare qualsiasi istanza della classe,
// Widget.widgetCount è zero (0)
trace("Conteggio iniziale Widget: " + Widget.widgetCount); // 0
var widget1:Widget = new Widget(); // 1
var widget2:Widget = new Widget(); // 2
var widget3:Widget = new Widget(); // 3
trace("Conteggio finale Widget: " + Widget.widgetCount); // 3
```

- 6.** Salvare le modifiche a `widget_test.fla`.

- 7.** Selezionare Controllo > Prova filmato per provare il file.

Nel pannello Output verranno visualizzate le informazioni seguenti:

```
Conteggio iniziale Widget: 0
Creazione widget # 1
Creazione widget # 2
Creazione widget # 3
Conteggio finale Widget: 3
```

## Informazioni sui metodi getter e setter

I metodi getter e setter sono metodi supplementari ovvero un'interfaccia pubblica per la modifica di membri di classe privati. Vengono utilizzati per definire una proprietà. L'accesso ai metodi getter e setter avviene come proprietà all'esterno della classe, anche se vengono definiti all'interno della classe come metodi. Le proprietà all'esterno della classe possono avere nomi diversi rispetto alla proprietà presente nella classe.

L'uso dei metodi getter e setter presenta alcuni vantaggi, ad esempio la possibilità di creare membri con funzionalità sofisticate accessibili come proprietà e di creare proprietà di sola lettura e sola scrittura.

Sebbene l'uso di questi metodi sia utile, si consiglia di non *esagerare* perché, in alcune situazioni, la gestione del codice potrebbe risultare più difficile. Essi forniscono inoltre accesso all'implementazione della classe, come membri pubblici, e nella programmazione orientata agli oggetti l'accesso diretto alle proprietà all'interno di una classe deve essere evitato.

È sempre consigliabile creare il maggior numero possibile di variabili di istanza private nelle classi e aggiungere metodi getter e setter di conseguenza perché spesso gli utenti non devono essere in grado di modificare determinate variabili delle classi. Se, ad esempio, è presente un metodo statico privato che tiene traccia del numero di istanze create per una classe specifica, un utente non deve essere in grado di modificare il contatore tramite codice, ma solo l'istruzione della funzione di costruzione deve poter incrementare la variabile ogni volta che viene chiamata. In una situazione di questo tipo è possibile creare una variabile di istanza privata e un metodo getter solo per la variabile del contatore. In questo modo gli utenti saranno in grado di recuperare il valore corrente solo tramite il metodo getter e non potranno impostare nuovi valori tramite il metodo setter. La creazione di un getter senza un setter rappresenta un metodo semplice per rendere determinate variabili della classe di sola lettura.

## Uso di metodi getter e setter

La sintassi dei metodi getter e setter è la seguente:

- Un metodo getter non accetta parametri e restituisce sempre un valore.
- Un metodo setter accetta sempre un parametro e non restituisce mai valori.

Generalmente le classi definiscono metodi getter che forniscono accesso in lettura e metodi setter che forniscono accesso in scrittura a una data proprietà. Ad esempio, nel caso in cui esista una classe che contiene un proprietà denominata `userName`:

```
private var userName:String;
```

Anziché consentire alle istanze della classe di accedere direttamente a questa proprietà (`user.userName = "Buster"`, ad esempio), la classe potrebbe disporre di due metodi, `getUserName()` e `setUserName()`, che verrebbero implementati come illustrato nell'esempio seguente.

### Per usare i metodi getter e setter:

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Immettere il codice seguente nella finestra Script:

```
class Login {
 private var __username:String;
 public function Login(username:String) {
 this.__username = username;
 }
 public function getUserName():String {
 return this.__username;
 }
 public function setUserName(value:String):Void {
 this.__username = value;
 }
}
```

**3.** Salvare il documento ActionScript come **Login.as**.

Come si può vedere, `getUserName()` restituisce il valore corrente di `userName` e `setUserName()` imposta il valore di `userName` sul parametro stringa passato al metodo.

**4.** Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA e salvarlo come **login\_test.fla** nella stessa directory del file Login.as.

**5.** Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var user:Login = new Login("RickyM");

// Chiama il metodo getUserName()
var userName:String = user.getUserName();
trace(userName); // RickyM

// Chiama il metodo setUserName()
user.setUserName("EnriqueI");
trace(user.getUserName()); // EnriqueI
```

**6.** Selezionare Controllo > Prova filmato per provare il file.

Nel pannello Output verranno visualizzate le informazioni seguenti:

RickyM  
EnriqueI

Se tuttavia si desidera utilizzare una sintassi più concisa, è possibile utilizzare metodi getter e setter impliciti che consentono di accedere direttamente alle proprietà di classe, scrivendo allo stesso tempo codice efficiente e formalmente corretto.

Per definire tali metodi, utilizzare gli attributi dei metodi `get` e `set`. I metodi creati ottengono o impostano il valore di una proprietà e aggiungono la parola chiave `get` o `set` prima del nome del metodo, come nell'esempio seguente:



I metodi getter/setter impliciti rappresentano una sintassi abbreviata rispetto al metodo `Object.addProperty()` di ActionScript 1.0.

**Per usare i metodi getter e setter impliciti:**

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.

2. Immettere il codice seguente nella finestra Script:

```
class Login2 {
 private var __username:String;
 public function Login2(username:String) {
 this.__username = username;
 }
 public function get userName():String {
 return this.__username;
 }
 public function set userName(value:String):Void {
 this.__username = value;
 }
}
```

3. Salvare il documento ActionScript come **Login2.as**.

Un metodo getter non accetta alcun parametro. Un metodo setter deve accettare un solo parametro obbligatorio e può avere lo stesso nome del metodo getter all'interno della stessa area di validità. I metodi getter e setter non possono avere lo stesso nome di altre proprietà. Nel codice di esempio precedente, ad esempio, in cui vengono definiti metodi getter e setter denominati `userName`, non è possibile inserire nella stessa classe una proprietà denominata `userName`.

4. Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA e salvarlo come **login2\_test.fla** nella stessa directory del file Login2.as.

5. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var user:Login2 = new Login2("RickyM");

// Viene chiamato il metodo "get"
var userNameStr:String = user.userName;
trace(userNameStr); // RickyM

// Viene chiamato il metodo "set"
user.userName = "EnriqueI";
trace(user.userName); // EnriqueI
```

Contrariamente ai metodi standard, i metodi getter e setter vengono richiamati senza parentesi né argomenti. I metodi getter e setter vengono richiamati quando si desidera una proprietà con lo stesso nome.

- 6.** Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file.

Nel pannello Output verranno visualizzate le informazioni seguenti:

RickyM  
EnriqueI

**NOTA**

Gli attributi dei metodi getter e setter non possono essere utilizzati nelle dichiarazioni dei metodi di interfaccia.

## Informazioni sulle classi dinamiche

Aggiungendo la parola chiave `dynamic` alla definizione di una classe, si specifica che gli oggetti basati sulla classe specificata possono aggiungere proprietà dinamiche e accedervi in fase di runtime. Creare classi dinamiche solo se sono effettivamente necessarie.

La verifica del tipo eseguita sulle classi dinamiche è meno rigorosa di quella eseguita sulle classi non dinamiche, in quanto i membri cui si accede all'interno della definizione di classe e nelle istanze di classe non vengono confrontati con quelli definiti nell'area di validità della classe. Le funzioni dei membri di classe, tuttavia, possono essere verificate per quanto riguarda i tipi restituiti e i tipi di parametro.

Per informazioni sulla creazione di classi dinamiche, vedere “[Creazione di classi dinamiche](#)” a pagina 233.

## Creazione di classi dinamiche

Per impostazione predefinita, le proprietà e i metodi di una classe sono fissi, ossia un'istanza di una classe non può creare proprietà o metodi non originariamente dichiarati o definiti dalla classe, né accedervi. Si consideri, ad esempio, una classe Person che definisce due proprietà: `userName` e `age`.

### Per creare una classe non dinamica:

**1.** Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.

**2.** Immettere il codice ActionScript seguente nella finestra Script:

```
class Person {
 public var userName:String;
 public var age:Number;
}
```

Se in un altro script viene creata un'istanza della classe Person e si desidera accedere alla proprietà della classe non esistente, il compilatore genera un errore.

**3.** Salvare il file sul disco rigido come `Person.as`.

4. Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA, quindi fare clic su OK.
5. Selezionare File > Salva con nome, assegnare al file il nome **person\_test.fla** e salvarlo nella stessa directory della classe Person creata in precedenza.
6. Aggiungere il codice seguente per creare una nuova istanza della classe Person (`firstPerson`) e cercare di assegnare un valore a una proprietà denominata `hairColor` che non esiste in questa classe:

```
var firstPerson:Person = new Person();
firstPerson.hairColor = "blue"; // Errore. Non è presente alcuna
proprietà denominata "hairColor".
```

7. Salvare il documento Flash.
8. Selezionare Controllo > Prova filmato per provare il codice.

Questo codice determina un errore del compilatore poiché la classe Person non dichiara alcuna proprietà denominata `hairColor`. Nella maggior parte dei casi, questo è effettivamente il risultato previsto. Sebbene gli errori del compilatore possano non essere bene accetti, sono in realtà molto utili per i programmatore perché, se creati in modo funzionale, aiutano a scrivere codice corretto, in quanto segnalano gli errori in una delle prime fasi del processo di scrittura del codice.

In alcuni casi, tuttavia, può essere necessario aggiungere in fase di runtime proprietà o metodi di una classe che non sono presenti nella definizione originale della classe e accedervi. A questo scopo è possibile utilizzare il modificatore di classe `dynamic`.

#### Per creare una classe dinamica:

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Selezionare File > Salva con nome e assegnare al file il nome **Person2.as**. Salvare il file sul disco rigido.
3. Immettere il codice seguente nella finestra Script:

```
dynamic class Person2 {
 public var userName:String;
 public var age:Number;
}
```

Questo codice ActionScript aggiunge la parola chiave `dynamic` alla classe Person dell'esempio precedente. Le istanze della classe Person2 possono aggiungere proprietà e metodi che non sono definiti nella classe originale e accedervi.

4. Salvare le modifiche apportate al file ActionScript.
5. Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA, quindi fare clic su OK.

6. Selezionare File > Salva con nome e assegnare al file il nome **person2\_test.fla**. Salvarlo nella stessa directory di Person2.as.
7. Aggiungere il codice seguente per creare una nuova istanza della classe Person2 (firstPerson) e assegnare un valore a una proprietà denominata hairColor che non esiste in questa classe:

```
var firstPerson:Person2 = new Person2();
firstPerson.hairColor = "blu";
trace(firstPerson.hairColor); // blu
```

8. Salvare le modifiche apportate al file person2\_test.fla.
9. Selezionare Controllo > Prova filmato per provare il codice.

Dato che la classe Flash personalizzata è dinamica, è possibile aggiungere metodi e proprietà alla classe in fase di runtime, ovvero quando viene riprodotto il file SWF. Quando si prova il codice, viene visualizzato il testo `blu` nel pannello Output.

Creare classi dinamiche solo se strettamente necessario per lo sviluppo dell'applicazione. La verifica del tipo eseguita sulle classi dinamiche è infatti meno rigorosa di quella eseguita sulle classi non dinamiche, in quanto i membri cui si accede all'interno della definizione di classe e nelle istanze di classe non vengono confrontati con quelli definiti nell'area di validità della classe. Le funzioni dei membri di classe, tuttavia, possono essere verificate per quanto riguarda i tipi restituiti e i tipi di parametro.

Le sottoclassi delle classi dinamiche sono a loro volta dinamiche, con una eccezione: le sottoclassi della classe MovieClip non sono dinamiche per impostazione predefinita, anche se la classe MovieClip è dinamica. Questa implementazione garantisce un maggior controllo sulle sottoclassi della classe MovieClip, perché è possibile scegliere se rendere o meno dinamiche le sottoclassi:

```
class A extends MovieClip {} // A non è dinamica
dynamic class B extends A {} // B è dinamica
class C extends B {} // C è dinamica
class D extends A {} // D non è dinamica
dynamic class E extends MovieClip{} // E è dinamica
```

Per informazioni sulle sottoclassi, vedere [Capitolo 7, “Ereditarietà” a pagina 279](#).

## Informazioni sull'incapsulamento

Nella progettazione orientata agli oggetti, gli oggetti sono considerati scatole nere che contengono o *incapsulano* funzionalità. Un programmatore dovrebbe essere in grado di interagire con un oggetto conoscendone solo le proprietà, i metodi e gli eventi (l'interfaccia di programmazione), ma senza conoscerne i dettagli di implementazione. Questo approccio consente di programmare a un livello di astrazione superiore e garantisce una struttura organizzativa per la creazione di sistemi complessi.

Per garantire l'incapsulamento, ActionScript 2.0 comprende, ad esempio, il controllo dell'accesso dei membri, affinché i dettagli dell'implementazione possano essere resi privati e invisibili al codice esterno a un oggetto che deve interagire con l'interfaccia di programmazione dell'oggetto, anziché con i relativi dati di implementazione. Questo approccio garantisce alcuni vantaggi importanti: consente ad esempio al creatore dell'oggetto di modificare l'implementazione dell'oggetto senza dover apportare modifiche al codice esterno all'oggetto, a condizione che l'interfaccia di programmazione non cambi.

Rendendo, ad esempio, tutte le variabili membro e di classe private e obbligando i programmatore che implementano le classi personalizzate ad accedere a tali variabili tramite metodi getter e setter si eseguirebbe un encapsulamento. In questo modo, se in futuro fosse necessario modificare la struttura delle variabili, sarebbe sufficiente modificare il comportamento delle funzioni getter e setter, anziché obbligare ogni sviluppatore a modificare il metodo di accesso alle variabili della classe.

Il codice seguente illustra come modificare la classe Person rispetto agli esempi precedenti, impostarne i membri di istanza come privati e definire metodi getter e setter per i membri di istanza privati:

```
class Person {
 private var __userName:String;
 private var __age:Number;
 public function get userName():String {
 return this.__userName;
 }
 public function set userName(value:String):Void {
 this.__userName = value;
 }
 public function get age():Number {
 return this.__age;
 }
 public function set age(value:Number):Void {
 this.__age = value;
 }
}
```

## Informazioni sull'uso della parola chiave this nelle classi

Utilizzare la parola chiave `this` come prefisso all'interno delle classi per i metodi e le variabili di membro. Sebbene non sia strettamente necessaria, la parola chiave `this` anteposta a una proprietà o a un metodo consente di comprendere se la proprietà o il metodo appartiene a una classe. In caso contrario, non è possibile determinare se la proprietà o il metodo appartiene alla superclasse.

È inoltre possibile utilizzare come prefisso il nome di una classe per le variabili statiche e i metodi anche all'interno di una classe per qualificare in modo migliore i riferimenti e rendere il codice più leggibile. A seconda dell'ambiente di scrittura del codice in uso, l'aggiunta di prefissi potrebbe inoltre attivare i suggerimenti sul codice.

### NOTA

L'aggiunta dei prefissi non è obbligatoria e molti sviluppatori non la ritengono necessaria, tuttavia Macromedia consiglia di aggiungere la parola chiave `this` come prefisso, perché permette di migliorare la leggibilità e di scrivere codice chiaro, indicando il contesto per metodi e variabili.

## Esempio: creazione di classi personalizzate

Dopo le nozioni di base relative a un file di classe e l'indicazione dei tipi di elementi che possono essere contenuti, vengono ora fornite linee guida generali per la creazione di un file di classe. Il primo esempio di questo capitolo dimostra come scrivere classi e inserirle in pacchetti, mentre il secondo esempio mostra come utilizzare i file di classe con un file FLA.

### ATTENZIONE

Il codice ActionScript nei file esterni viene compilato in un file SWF al momento della pubblicazione, dell'esportazione, della verifica o del debug di un file FLA. Pertanto, se si apportano modifiche a un file esterno, salvare il file e ricompilare gli eventuali file FLA che lo utilizzano.

Come illustrato in “[Creazione di file di classi personalizzate](#)” a pagina 208, una classe è costituita da due parti principali: la *dichiarazione* e il *corpo*. In sostanza, la dichiarazione della classe consiste nell'istruzione `class`, seguita da un identificatore relativo al nome della classe e quindi da una parentesi graffa aperta e una chiusa (`{}`). I dati contenuti all'interno delle parentesi graffe rappresentano il corpo della classe, come nell'esempio seguente:

```
class className {
 // Corpo della classe
}
```

Si ricordi che: è possibile definire le classi solo in file di ActionScript esterni. Non è possibile, ad esempio, definire una classe nello script di un fotogramma in un file FLA. Per questo esempio, pertanto, occorre creare un nuovo file.

Nella forma più basilare, una dichiarazione consiste nella parola chiave `class`, seguita dal nome della classe (Person, in questo caso) e quindi da parentesi graffe a sinistra e a destra (`{}`). Tutti i dati presenti all'interno delle parentesi sono detti corpo della classe; in questo punto vengono definiti i metodi e le proprietà della classe.

Al termine dell'esempio, l'ordine delle informazioni di base nei file di classe sarà il seguente:

- Commenti di documentazione
- Dichiarazione di classe
- Funzione di costruzione
- Corpo della classe

In questo capitolo non viene illustrata la creazione di sottoclassi. Per ulteriori informazioni sull'ereditarietà e la creazione di sottoclassi, consultare il [Capitolo 7, “Ereditarietà” a pagina 279](#).

In questo esempio sono presentati i seguenti argomenti:

- “[Informazioni sulle linee guida generali per la creazione di una classe](#)” a pagina 239
- “[Creazione di file di classe e inserimento in pacchetti](#)” a pagina 240
- “[Creazione della funzione di costruzione](#)” a pagina 243
- “[Aggiunta di metodi e proprietà](#)” a pagina 245
- “[Controllo dell'accesso dei membri delle classi](#)” a pagina 248
- “[Documentazione delle classi](#)” a pagina 250

Un file di esempio sul disco rigido dimostra come creare un menu dinamico con dati XML e un file di classe personalizzato. Nell'esempio è presente una chiamata alla funzione di costruzione `XmlMenu()` di ActionScript, alla quale vengono passati due parametri: il percorso al file di menu XML e un riferimento alla linea temporale corrente. Il resto delle funzionalità risiede in un file di classe personalizzato, `XmlMenu.as`.

È possibile trovare il file di origine di esempio, `xmlmenu.fla`, nella cartella Samples sul disco rigido.

- In Windows, posizionarsi in `unità di avvio:\Programmi\Macromedia\Flash 8\Samples e Tutorials\Samples\ActionScript\XML_Menu`.
- In Macintosh: `Macintosh HD/Applicazioni/Macromedia Flash 8/Samples e Tutorials/Samples\ActionScript/XML_Menu`.

## Informazioni sulle linee guida generali per la creazione di una classe

Per la creazione di file di classi personalizzate si consiglia di attenersi alle linee guida presentate di seguito che aiutano a scrivere classi corrette dal punto di vista del funzionamento e del formato. Ci si eserciterà con queste linee guida negli esempi seguenti.

- In generale è buona norma inserire una sola dichiarazione per riga e non dichiarazioni dello stesso tipo o di tipi diversi sulla stessa riga; formattare le dichiarazioni come illustrato nell'esempio seguente:

```
private var SKU:Number; // Numero di identificazione del prodotto (SKU)
private var quantity:Number; // Quantità di prodotto
```

- Inizializzare le variabili locali quando vengono dichiarate se il valore iniziale non è determinato da un calcolo. Per informazioni sull'inizializzazione di variabili, vedere ["Aggiunta di metodi e proprietà" a pagina 245](#).

- Dichiare le variabili prima di utilizzarle, inclusi i cicli. Il codice seguente, ad esempio, dichiara la variabile dell'iteratore del ciclo (*i*) prima di utilizzarla nel ciclo *for*:

```
var my_array:Array = new Array("uno", "due", "tre");
var i:Number;
for (i = 0 ; i < my_array.length; i++) {
 trace(i + " = " + my_array[i]);
}
```

- Non utilizzare dichiarazioni locali che nascondono dichiarazioni di livello superiore, ad esempio non dichiarare una variabile due volte, come illustrato nell'esempio seguente:

```
// Codice scorretto
var counter:Number = 0;
function myMethod() {
 var counter:Number;
 for (counter = 0; counter <= 4; counter++) {
 // Istruzioni;
 }
}
```

Il codice dichiara la stessa variabile all'interno di un blocco interno.

- Non assegnare molte variabili a un solo valore in un'istruzione perché il codice risulterebbe difficile da leggere, come illustrato nel codice ActionScript seguente:

```
// Formato scorretto
xPos = yPos = 15;

or

// Formato scorretto
class User {
 private var m_username:String, m_password:String;
}
```

- Non utilizzare variabili di istanza pubbliche o variabili statiche, di classe o membro pubbliche se non è assolutamente necessario e se non si è prima verificato che siano state dichiarate pubbliche in modo esplicito.
- Impostare la maggior parte delle variabili membro come private, a meno che non esista un motivo per renderle pubbliche. Dal punto di vista della progettazione è decisamente preferibile impostare le variabili di membro come private e consentirvi l'accesso solo tramite un gruppo limitato di funzioni getter e setter.

## Informazioni sull'assegnazione di nomi ai file di classe

I nomi delle classi devono essere identificatori, ovvero il primo carattere deve essere una lettera, un carattere di sottolineatura (\_) o un simbolo di dollaro (\$) e i caratteri seguenti devono essere lettere, numeri, caratteri di sottolineatura o simboli di dollaro. È preferibile utilizzare solo lettere nei nomi di classe.

Inoltre, il nome della classe specificato deve corrispondere esattamente (maiuscole e minuscole comprese) a quello del file ActionScript che lo contiene. Nell'esempio seguente, se si crea una classe denominata Rock, il file ActionScript che contiene la definizione della classe deve essere denominato Rock.as:

```
// Nel file Rock.as
class Rock {
 // Corpo della classe Rock
}
```

Nella sezione seguente viene creata e denominata una definizione di classe. Per creare file di classe, assegnarvi un nome e inserirli in pacchetti, consultare la sezione “[Creazione di file di classe e inserimento in pacchetti](#)” a pagina 240. Per ulteriori informazioni sull'assegnazione di nomi ai file di classe, consultare il “[Assegnazione di nomi a classi e oggetti](#)” a pagina 800.

## Creazione di file di classe e inserimento in pacchetti

In questa sezione vengono creati file di classe per questo esempio (“[Esempio: creazione di classi personalizzate](#)” a pagina 237), vi si assegna un nome e i file vengono inseriti in pacchetti. Le sezioni seguenti mostrano come scrivere file di classe completi, ma semplici. Per informazioni dettagliate sui pacchetti, vedere “[Informazioni sui pacchetti](#)” a pagina 200, “[Confronto tra classi e pacchetti](#)” a pagina 201 e “[Operazioni con i pacchetti](#)” a pagina 202.

Per creare un file di classe è necessario decidere dove memorizzarlo. Nella procedura che segue, per motivi di semplicità, il file di classe e il file FLA dell'applicazione che utilizza il file di classe vengono salvati nella stessa directory. Tuttavia, se si desidera verificare la sintassi, è necessario specificare in Flash la posizione del file. In genere, quando si crea un'applicazione, si aggiunge la directory in cui sono memorizzati l'applicazione e i file di classe al percorso di classe di Flash. Per informazioni sui percorsi di classe, vedere [“Informazioni sull'impostazione e la modifica del percorso di classe”](#) a pagina 214.

I file di classe sono detti anche file ActionScript (AS) e vengono creati nello strumento di modifica di Flash oppure utilizzando un editor esterno, come Macromedia Dreamweaver e Macromedia Flex Builder.

**NOTA**

Il nome di una classe (ClassA) deve corrispondere esattamente al nome del file AS che contiene la classe (ClassA.as). Questa condizione è di estrema importanza: se i due nomi non corrispondono, ad esempio perché vengono specificate maiuscole e minuscole diverse, la classe non viene compilata.

**Per creare un file di classe e una dichiarazione di classe:**

1. Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo documento FLA, quindi fare clic su OK.
2. Selezionare File > Salva con nome, assegnare al nuovo file il nome **package\_test.fla** e salvare il documento Flash nella directory corrente.  
In un passaggio successivo si aggiungerà contenuto a questo documento.
3. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
4. Selezionare File > Salva con nome e creare una nuova sottodirectory denominata **com**; quindi eseguire le seguenti operazioni:
  - a. Nella sottodirectory com, creare una nuova sottodirectory denominata **macromedia**.
  - b. Nella sottodirectory macromedia, creare una nuova sottodirectory denominata **utils**.
  - c. Salvare il documento ActionScript corrente nella directory utils e assegnarvi il nome **ClassA.as**.

5. Immettere il codice seguente nella finestra Script:

```
class com.macromedia.utils.ClassA {
}
```

Il codice precedente crea una nuova classe di nome ClassA nel pacchetto com.macromedia.utils.

6. Salvare il documento ActionScript ClassA.as.
7. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.

**8.** Selezionare File > Salva con nome, assegnare al file il nome **ClassB.as**, salvarlo nella stessa directory del file ClassA.as creato al punto precedente.

**9.** Immettere il codice seguente nella finestra Script:

```
class com.macromedia.utils.ClassB {
}
```

Il codice precedente crea una nuova classe di nome ClassB nel pacchetto com.macromedia.utils.

**10.** Salvare le modifiche apportate sia al file ClassA.as che al file ClassB.as.

I file di classe utilizzati in un file FLA vengono importati in un file SWF alla compilazione. Il codice di un file di classe deve seguire una determinata metodologia e un certo ordine, come illustrato nelle sezioni seguenti.

Se si creano più classi personalizzate, utilizzare pacchetti per organizzare i file di classe. Un pacchetto è una directory che contiene uno o più file di classe ed è contenuto, a sua volta, in una directory di un percorso di classe. I nomi di classe devono essere completi e contenere il nome del file in cui la classe viene dichiarata, ovvero la directory (pacchetto) in cui la classe è contenuta. Per ulteriori informazioni sui percorsi di classe, vedere “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214.

Una classe denominata com.macromedia.docs.YourClass, ad esempio, viene memorizzata nella directory com/macromedia/docs. La dichiarazione della classe nel file YourClass.as è analoga alla seguente:

```
class com.macromedia.docs.YourClass {
 // La classe creata
}
```

**NOTA**

Nella sezione seguente, “[Esempio: creazione di classi personalizzate](#)” a pagina 237, si scriverà la dichiarazione di classe per riflettere la directory del pacchetto.

Per questo motivo, è consigliabile pianificare la struttura del pacchetto prima di iniziare a creare le classi. Altrimenti, se si decide di spostare i file di classe dopo averli creati, occorre modificare le istruzioni di dichiarazione delle classi affinché riflettano la nuova posizione.

#### Per inserire i file di classe in un pacchetto:

**1.** Definire il nome di pacchetto che si desidera utilizzare.

I nomi di pacchetto devono essere intuitivi e facilmente identificabili anche da altri sviluppatori. Tenere a mente che il nome di pacchetto corrisponde anche a una struttura di directory specifica. Ad esempio, le classi nel pacchetto com.macromedia.utils dovranno essere inserite nella cartella com/macromedia/utils sul disco rigido.

- 2.** Dopo aver scelto un nome per il pacchetto, creare la necessaria struttura di directory.

Ad esempio, se al pacchetto è stato assegnato il nome com.macromedia.utils, è necessario creare la struttura di directory com/macromedia/utils e inserire le classi nella cartella utils.

- 3.** Usare il prefisso com.macromedia.utils per tutte le classi create in questo pacchetto.  
ad esempio se il nome della classe è ClassA, all'interno del file di classe com/macromedia/utils/ClassA.as il nome completo della classe è com.macromedia.utils.ClassA.
- 4.** Se in futuro si modifica la struttura del pacchetto, è necessario modificare non solo la struttura delle directory, ma anche il nome del pacchetto all'interno di ogni file di classe, nonché ogni istruzione import o riferimento a una classe all'interno di tale pacchetto.

Per ulteriori indicazioni sulla scrittura di file di classe, vedere “[Creazione della funzione di costruzione](#)” a pagina 243.

## Creazione della funzione di costruzione

In “[Creazione di file di classe e inserimento in pacchetti](#)” a pagina 240 è stato illustrato come creare la dichiarazione di classe. In questa parte del capitolo, viene creata la cosiddetta *funzione di costruzione* del file di classe.

**NOTA**

La creazione di commenti, istruzioni e dichiarazioni è illustrata nelle sezioni seguenti.

Per funzioni di costruzione si intendono funzioni che consentono di inizializzare (*definire*) le proprietà e i metodi di una classe. Per definizione, le funzioni di costruzione sono funzioni incluse in una definizione di classe, con la stessa denominazione della classe. Il codice seguente, ad esempio, definisce una classe Person e implementa una funzione di costruzione. Nella programmazione orientata agli oggetti, la funzione di costruzione inizializza ogni nuova istanza di una classe.

La funzione di costruzione di una classe è una funzione speciale che viene chiamata automaticamente quando si crea un'istanza di una classe utilizzando l'operatore new. e presenta lo stesso nome della classe che la contiene. La classe Person creata in precedenza, ad esempio, conteneva la seguente funzione di costruzione:

```
// Funzione di costruzione della classe Person
public function Person (uname:String, age:Number) {
 this.__name = uname;
 this.__age = age;
}
```

Per la creazione di funzioni di costruzione, considerare quanto segue:

- Se nessuna funzione di costruzione viene dichiarata esplicitamente, ossia non viene creata una funzione il cui nome corrisponde a quello della classe, il compilatore crea automaticamente una funzione di costruzione vuota.
- Una classe può contenere solo una funzione di costruzione; in ActionScript 2.0 le funzioni di costruzione multiple non sono consentite.
- Una funzione di costruzione non può avere un tipo restituito.

In genere il termine *funzione di costruzione* viene inoltre utilizzato quando si crea un oggetto (ovvero se ne creano istanze) in base a una classe specifica. Le istruzioni seguenti sono chiamate a funzioni di costruzione della classe Array di primo livello e della classe personalizzata Person:

```
var day_array:Array = new Array("Dom", "Lun", "Mar", "Mer", "Gio", "Ven",
 "Sab");
var somePerson:Person = new Person("Paolo", 30);
```

A questo punto si aggiunge una funzione speciale, detta funzione di costruzione.

**NOTA**

L'esercizio seguente è contenuto in “[Esempio: creazione di classi personalizzate](#)” [a pagina 237](#). Se non si desidera proseguire con l'esempio, è possibile scaricare i file di classe da [www.helpexamples.com/flash/learnas/classes/](http://www.helpexamples.com/flash/learnas/classes/).

### Per aggiungere le funzioni di costruzione ai file di classe:

1. Aprire il file di classe ClassA.as nello strumento di creazione di Flash.
2. Modificare il file di classe esistente aggiungendo il codice riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassA {
 function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
}
```

Il codice riportato sopra definisce un metodo della funzione di costruzione per la classe ClassA. La funzione di costruzione traccia una stringa semplice e la visualizza nel pannello Output per segnalare quando viene creata una nuova istanza della classe.

3. Aprire il file di classe ClassB.as nello strumento di creazione di Flash.
4. Modificare il file di classe aggiungendo il codice riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassB {
 function ClassB() {
 trace("funzione di costruzione di ClassB");
 }
}
```

5. Prima di continuare, salvare entrambi i file ActionScript.

Per ulteriori indicazioni sulla scrittura del file di classe, vedere “[Aggiunta di metodi e proprietà](#)” a pagina 245.

## Aggiunta di metodi e proprietà

Per creare le proprietà per le classi ClassA e ClassB, definire le variabili utilizzando la parola chiave var.

**NOTA**

I tre esercizi seguenti sono contenuti in “[Esempio: creazione di classi personalizzate](#)” a pagina 237. Se non si desidera proseguire con l'esempio, è possibile scaricare i file di classe da [www.helplexamples.com/flash/learnas/classes/](http://www.helplexamples.com/flash/learnas/classes/).

### Per aggiungere proprietà alle classi ClassA e ClassB:

1. Aprire ClassA.as e ClassB.as nello strumento di creazione di Flash.
2. Modificare il file ActionScript ClassA.as aggiungendo il codice riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassA {
 static var _className:String;
 function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
}
```

Il blocco di codice riportato sopra aggiunge una sola nuova variabile statica, \_className che contiene il nome della classe corrente.

3. Modificare la classe ClassB aggiungendo la variabile statica affinché il codice risulti analogo a quello riportato sopra.
4. Prima di continuare, salvare entrambi i file ActionScript.

**SUGGERIMENTO**

Per convenzione, le proprietà di classe vengono definite all'inizio del corpo della classe. Sebbene si possa anche fare altrimenti, in questo modo il codice risulta più semplice da comprendere.

Nelle dichiarazioni delle variabili si utilizza la sintassi che segue i due punti, ad esempio `var username:String` e `var age:Number`. Questo è un esempio di tipizzazione forte dei dati. Quando si assegna un tipo a una variabile utilizzando il formato `var nomeVariabile:tipoVariabile`, il compilatore di ActionScript assicura che qualsiasi valore assegnato alla variabile corrisponda al tipo specificato. Se nel file FLA in cui viene importata la classe non viene utilizzato il tipo di dati corretto, il compilatore genera un errore. Per ulteriori informazioni sulla tipizzazione forte dei dati, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

I membri di una classe sono rappresentati da proprietà (dichiarazioni di variabili) e metodi (dichiarazioni di funzioni). Tutte le proprietà e tutti i metodi devono essere dichiarati e definiti all'interno del corpo della classe (le parentesi graffe [{}]). In caso contrario, si verificherà un errore durante la compilazione. Per informazioni sui membri, vedere “[Informazioni su metodi e proprietà \(membri\) pubblici, privati e statici](#)” a pagina 220.

### Per aggiungere metodi alle classi ClassA e ClassB:

1. Aprire ClassA.as e ClassB.as nello strumento di creazione di Flash.
2. Modificare il file della classe ClassA aggiungendo il codice riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassA {
 static var _className:String;

 function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
 function doSomething():Void {
 trace("ClassA - doSomething()");
 }
}
```

Il blocco di codice in grassetto crea un nuovo metodo nella classe che traccia una stringa e la visualizza nel pannello Output.

3. In ClassA.as, selezionare Strumenti > Controlla sintassi per controllare la sintassi del file ActionScript.

Se vengono rilevati errori nel pannello Output, confrontare il codice ActionScript dello script con il codice completo creato al punto precedente. Se non si riesce a risolvere gli errori del codice, prima di continuare, copiare e incollare il codice completo nella finestra Script.

**4.** Controllare la sintassi di ClassB.as come è stato fatto per ClassA.as.

Se nel pannello Output vengono visualizzati errori, prima di continuare, copiare e incollare il codice completo nella finestra Script:

```
class com.macromedia.utils.ClassB {
 static var _className:String;

 function ClassB() {
 trace("funzione di costruzione di ClassB");
 }
 function doSomething():Void {
 trace("ClassB - doSomething()");
 }
}
```

**5.** Prima di continuare, salvare entrambi i file ActionScript.

Le proprietà possono essere inizializzate inline, ovvero quando vengono dichiarate, con valori predefiniti, come nell'esempio seguente:

```
class Person {
 var age:Number = 50;
 var username:String = "Giovanni Bianchi";
}
```

Quando si inizializzano proprietà inline, l'espressione a destra di un'assegnazione deve essere una costante della fase di compilazione, vale a dire che l'espressione non può indicare alcun dato impostato o definito nella fase di runtime. Le costanti di compilazione includono caratteri letterali di stringa, numeri, valori booleani, null e undefined, oltre alle funzioni di costruzione per le seguenti classi di primo livello: Array, Boolean, Number, Object e String.

**Per inizializzare proprietà inline:**

1. Aprire ClassA.as e ClassB.as nello strumento di creazione di Flash.
2. Modificare il file della classe ClassA aggiungendo il codice ActionScript riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassA {
 static var _className:String = "ClassA";

 function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
 function doSomething():Void {
 trace("ClassA - doSomething()");
 }
}
```

L'unica differenza tra il file di classe esistente e il blocco di codice precedente è rappresentata dalla presenza di un valore definito per la variabile statica `_className`, ovvero "ClassA".

3. Modificare il file di classe ClassB aggiungendo la proprietà inline e modificando il valore su "ClassB".
4. Prima di continuare, salvare entrambi i file ActionScript.

Questa regola si applica solo alle variabili di istanza (variabili che vengono copiate in ciascuna istanza di una classe) e non alle variabili di classe (variabili che appartengono alla classe).



Quando si inizializzano array inline, per tutte le istanze della classe viene creato un solo array.

Per ulteriori indicazioni sulla scrittura del file di classe, vedere "[Controllo dell'accesso dei membri delle classi](#)" a pagina 248.

## Controllo dell'accesso dei membri delle classi

Per impostazione predefinita, qualsiasi proprietà o metodo di una classe risulta accessibile da qualsiasi altra classe: tutti i membri di una classe sono pubblici. Tuttavia, in alcuni casi, può essere necessario proteggere l'accesso ai dati o ai metodi di una classe da parte di altre classi. A questo scopo, occorre rendere tali membri privati, ossia disponibili solo per la classe in cui vengono dichiarati o definiti.

I membri pubblici o privati vengono specificati utilizzando l'attributo `public` o `private`. Ad esempio, il seguente codice dichiara una variabile privata (una proprietà) e un metodo privato (una funzione). La classe seguente (`LoginClass`) definisce una proprietà privata denominata `userName` e un metodo privato denominato `getUserName()`:

```
class LoginClass {
 private var userName:String;
 private function getUserName():String {
 return this.userName;
 }
 // Funzione di costruzione:
 public function LoginClass(user:String) {
 this.userName = user;
 }
}
```

I membri privati (proprietà e metodi) sono accessibili solo alla classe che definisce tali membri e alle relative sottoclassi. Le istanze della classe originale, o le istanze delle sottoclassi di quella classe, non possono accedere a proprietà e metodi dichiarati come privati; i membri privati, in altri termini, sono accessibili solo all'interno delle definizioni della classe, non a livello dell'istanza. Nell'esempio che segue, viene modificato l'accesso dei membri nei file di classe.

**NOTA**

L'esercizio è contenuto in "[Esempio: creazione di classi personalizzate](#)" a pagina 237. Se non si desidera proseguire con l'esempio, è possibile scaricare i file di classe da [www.helpexamples.com/flash/learnas/classes/](http://www.helpexamples.com/flash/learnas/classes/).

**Per controllare l'accesso dei membri:**

1. Aprire ClassA.as e ClassB.as nello strumento di creazione di Flash.
2. Modificare il file ActionScript ClassA.as aggiungendo il codice ActionScript riportato in grassetto, affinché corrisponda a quanto segue:

```
class com.macromedia.utils.ClassA {
 private static var _className:String = "ClassA";

 public function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
 public function doSomething():Void {
 trace("ClassA - doSomething()");
 }
}
```

Il codice riportato sopra imposta entrambi i metodi (la funzione di costruzione di ClassA e il metodo doSomething()) come pubblici. È possibile quindi accedervi da script esterni. La variabile \_className statica viene impostata come privata. È possibile quindi accedervi solo dall'interno della classe e non da script esterni.

3. Modificare il file ActionScript ClassB.as aggiungendo lo stesso accesso di metodi e proprietà della classe ClassA.
4. Prima di continuare, salvare entrambi i file ActionScript.

Un'istanza di ClassA o ClassB non può accedere a membri privati. Il codice seguente, ad esempio, aggiunto al fotogramma 1 della linea temporale di un file FLA provocherebbe un errore del compilatore, in base al quale il metodo verrebbe indicato come privato e non accessibile:

```
import com.macromedia.utils.ClassA;
var a:ClassA = new ClassA();
trace(a._className); // Errore. Il membro è privato e non accessibile.
```

Il controllo dell'accesso dei membri viene esercitato solo in fase di compilazione. In fase di runtime, Flash Player non distingue tra membri pubblici e privati.

Per ulteriori indicazioni sulla scrittura del file di classe, vedere “[Documentazione delle classi](#)” a pagina 250.

## Documentazione delle classi

I commenti nelle classi e nelle interfacce rappresentano una parte importante della documentazione per altri utenti. Se, ad esempio, si desidera distribuire i file di classe nella comunità Flash o si lavora in un team di designer o sviluppatori che utilizzano tali file di classe per il loro lavoro o, ancora, si collabora a un progetto con altri sviluppatori, la documentazione aiuta gli altri utenti a comprendere lo scopo e l'origine della classe.

In un file di interfaccia o di classe tipico sono presenti due tipi di commenti: *commenti di documentazione* e *commenti di implementazione*. I commenti di documentazione vengono utilizzati per descrivere le specifiche del codice, ma non l'implementazione. I commenti di implementazione, invece, vengono utilizzati per impostare porzioni di codice come commento o inserire commenti sull'implementazione di determinate sezioni di codice. Per i due tipi di commenti vengono utilizzati delimitatori leggermente diversi: i commenti di documentazione sono delimitati con `/**` e `*/`, mentre i commenti di implementazione sono delimitati con `/*` e `*/`.



I commenti di documentazione non sono un costrutto del linguaggio ActionScript 2.0. Tuttavia, rappresentano un modo comune per strutturare i commenti in un file di classe che possono essere utilizzati nei file AS.

Utilizzare i commenti di documentazione per descrivere interfacce, classi, metodi e funzioni di costruzione. Includere un unico commento di documentazione per classe, interfaccia o membro, inserendolo subito prima della dichiarazione.

Se si desidera inserire ulteriori informazioni di documentazione e lo spazio nei commenti di documentazione non è sufficiente, fare ricorso ai commenti di implementazione, utilizzando il formato dei commenti a blocchi o a riga singola, come descritto in “[Informazioni sui commenti](#)” a pagina 96. I commenti di implementazione, se necessari, vengono inseriti subito dopo la dichiarazione.



Non inserire commenti che non si riferiscano direttamente alla classe che viene letta, ad esempio commenti che descrivono il pacchetto corrispondente.



L'esercizio seguente è contenuto in “[Esempio: creazione di classi personalizzate](#)” a pagina 237. Se non si desidera proseguire con l'esempio, è possibile scaricare i file di classe da [www.helpexamples.com/flash/learnas/classes/](http://www.helpexamples.com/flash/learnas/classes/).

### **Per documentare i file di classe:**

1. Aprire ClassA.as e ClassB.as nello strumento di creazione di Flash.
2. Modificare il file della classe ClassA, aggiungendo il nuovo codice riportato in grassetto all'inizio del file della classe:

```
/**
 * classe ClassA
 * Versione 1.1
 * 6/21/2005
 * Copyright Macromedia, Inc.
 */
class com.macromedia.utils.ClassA {
 private static var _className:String = "ClassA";

 public function ClassA() {
 trace("funzione di costruzione di ClassA");
 }
 public function doSomething():Void {
 trace("ClassA - doSomething()");
 }
}
```

Il codice precedente ha aggiunto un commento all'inizio del file della classe. L'aggiunta di commenti ai file Flash e ActionScript rappresenta una pratica consigliata per aggiungere informazioni utili, ad esempio l'autore della classe, la data di modifica, le informazioni di copyright ed eventuali problemi o errori presenti nel file.

3. Aggiungere un commento analogo all'inizio del file ActionScript ClassB.as, modificando il nome della classe ed eventuali altri informazioni come appropriato.
4. Prima di continuare, salvare entrambi i file ActionScript.

È inoltre possibile aggiungere commenti a blocchi, a riga singola o finali all'interno del codice della classe. Per informazioni sulla scrittura di commenti corretti all'interno del codice, vedere “[Scrittura di commenti significativi](#)” a pagina 804. Per informazioni di tipo generale sui commenti, vedere “[Commenti a riga singola](#)” a pagina 97, “[Commenti su più righe](#)” a pagina 97 e “[Commenti finali](#)” a pagina 98.

Per informazioni sull'uso di questi file di classi personalizzate in un file SWF, vedere “[Esempio: uso di file di classi personalizzate in Flash](#)” a pagina 252.

# Esempio: uso di file di classi personalizzate in Flash

In questo esempio vengono utilizzati file di classe scritti nell'esempio denominato "["Esempio: creazione di classi personalizzate"](#) a pagina 237 oppure è possibile scaricarli da [www.helpexamples.com/flash/learnas/classes/](http://www.helpexamples.com/flash/learnas/classes/). Se è stato completato "["Esempio: creazione di classi personalizzate"](#) a pagina 237, trovare ClassA.as e ClassB.as sul disco rigido.

Dato che il nome di pacchetto del file di classe ClassA è `com.macromedia.utils.ClassA`, i file di classe devono essere salvati nella struttura di directory corrispondente. Creare una sottocartella denominata com nella directory corrente e, all'interno di questa cartella, aggiungere una nuova cartella denominata macromedia. Aggiungere una terza e ultima sottodirectory all'interno della cartella macromedia e assegnarvi il nome utils. Salvare i file di classe ClassA.as e ClassB.as all'interno della cartella utils. A questo punto è possibile iniziare l'esempio.

Le classi personalizzate create in "["Esempio: creazione di classi personalizzate"](#) a pagina 237 possono essere utilizzate con un file FLA. In questo esempio vengono utilizzate per creare una piccola applicazione in Flash. Le classi vengono compilate nel file SWF quando si pubblica il documento. Negli esercizi seguenti viene illustrato l'utilizzo dei percorsi di classe, l'uso dei file di classe nell'applicazione, come pure come importare classi e pacchetti.

Per continuare l'esercizio, passare a "["Importazione di classi e pacchetti"](#) a pagina 252.

## Importazione di classi e pacchetti

Per fare riferimento a una classe in un altro script, utilizzare il nome del pacchetto della classe come prefisso per il nome della classe. La combinazione tra il nome della classe e il percorso del relativo pacchetto rappresenta il nome completo della classe. Se una classe risiede in una directory di un percorso di classe di primo livello e non in una sottodirectory di una cartella di un percorso di classe, il relativo nome completo corrisponde semplicemente al nome della classe.

Quando si specificano i percorsi del pacchetto, utilizzare la notazione del punto (.) per separare i nomi di directory del pacchetto. I percorsi dei pacchetti sono gerarchici e ogni punto rappresenta una directory nidificata. Se, ad esempio, si crea una classe denominata `ClassName` che risiede nel pacchetto `com/macromedia/docs/learnAs2` nel percorso di classe, per creare un'istanza della classe è possibile specificarne il nome completo.

È possibile inoltre utilizzare il nome completo della classe per assegnare il tipo alle variabili, come nell'esempio seguente:

```
var myInstance:com.macromedia.docs.learnAs2.ClassName = new
 com.macromedia.docs.learnAs2.ClassName();
```

Se si utilizza l'istruzione `import` per importare i pacchetti in uno script, è possibile utilizzare il nome abbreviato della classe invece del nome completo. e il carattere jolly (\*) per importare tutte le classi contenute in un pacchetto. In questo caso, non occorre utilizzare il nome completo della classe ogni volta che si utilizza la classe.

Si supponga, ad esempio, che in uno script sia stata importata la classe riportata sopra con l'ausilio dell'istruzione `import`, come nell'esempio seguente:

```
import com.macromedia.docs.learnAs2.util.UserClass;
```

Successivamente, nello stesso script, è possibile fare riferimento alla classe utilizzandone il nome abbreviato, come illustrato nell'esempio seguente:

```
var myUser:UserClass = new UserClass();
```

È possibile utilizzare il carattere jolly (\*) per importare tutte le classi contenute in un determinato pacchetto. Se, ad esempio, si dispone di un pacchetto denominato `com.macromedia.utils` che contiene due file di classe ActionScript, `ClassA.as` e `ClassB.as`, in un altro script è possibile importare entrambe le classi di quel pacchetto utilizzando il carattere jolly, come illustrato di seguito:

```
import com.macromedia.utils.*;
```

L'esempio seguente mostra come fare riferimento direttamente a una delle due classi nello stesso script:

```
var myA:ClassA = new ClassA();
var myB:ClassB = new ClassB();
```

L'istruzione `import` è valida solo per lo script corrente (fotogramma o oggetto) da cui viene chiamata. Se una classe importata non viene utilizzata in uno script, non viene inclusa nel codice byte del file SWF risultante e non è disponibile per nessun file SWF che il file FLA contenente l'istruzione `import` può caricare.

**NOTA**

L'esercizio seguente fa parte di "[Esempio: uso di file di classi personalizzate in Flash](#)" [a pagina 252](#) che continua gli esempi "[Esempio: creazione di classi personalizzate](#)". Per utilizzare le classi `ClassA` e `ClassB`, è possibile scaricare i file di classe da [www.helpexamples.com/flash/learnas/classes/](http://www.helpexamples.com/flash/learnas/classes/).

### Per importare una classe o un pacchetto:

1. Aprire il file package\_test.fla.
2. Immettere il codice seguente nella finestra Script:

```
import com.macromedia.utils.*;
var a = new ClassA(); // funzione di costruzione della classe ClassA
var b = new ClassB(); // funzione di costruzione della classe ClassB
```

Il blocco di codice riportato sopra importa ognuna delle classi all'interno del pacchetto `com.macromedia.utils` utilizzando il carattere jolly (\*). Viene quindi creata una nuova istanza della classe ClassA in seguito alla quale il metodo della funzione di costruzione traccia un messaggio e lo visualizza nel pannello Output. Viene creata un'istanza anche della classe ClassB che a sua volta invia messaggi di debug nel pannello Output.

3. Prima di continuare, salvare le modifiche al documento Flash.

Per informazioni sull'uso di questi file di classe in un file Flash, vedere “[Creazione di istanze di classi in un esempio](#)” a pagina 254.

## Creazione di istanze di classi in un esempio

Le istanze sono oggetti che contengono tutte le proprietà e i metodi di una determinata classe. Gli array, ad esempio, sono istanze della classe Array ed è pertanto possibile utilizzare qualsiasi metodo o proprietà della classe Array con qualsiasi istanza di array. In alternativa è possibile creare una classe personalizzata, ad esempio UserSettings, e quindi creare un'istanza di questa classe.

Continuando dall'esempio creato in “[Esempio: uso di file di classi personalizzate in Flash](#)” a pagina 252, è stato modificato un file FLA per importare le classi scritte per evitare di farvi riferimento sempre tramite il nome completo.

Il passaggio successivo di questo esempio (“[Esempio: uso di file di classi personalizzate in Flash](#)” a pagina 252) consiste nel creare un'istanza delle classi ClassA e ClassB in uno script, ad esempio uno script di fotogramma in un documento Flash package\_test.fla, e assegnarvi una variabile. Per creare un'istanza di una classe personalizzata, si utilizza l'operatore new come durante la creazione di un'istanza di una classe ActionScript di primo livello (ad esempio la classe Date o Array). Per fare riferimento alla classe, utilizzare il nome completo della classe o importare la classe stessa, come illustrato in “[Importazione di classi e pacchetti](#)” a pagina 252.



L'esercizio seguente fa parte di “[Esempio: uso di file di classi personalizzate in Flash](#)” a pagina 252 che continua gli esempi “[Esempio: creazione di classi personalizzate](#)”.

**Per creare una nuova istanza delle classi ClassA e ClassB:**

1. Aprire il file `package_test.fla`.
2. Nella finestra Script immettere il codice riportato in grassetto di seguito:

```
import com.macromedia.utils.*;
var a:ClassA = new ClassA(); // funzione di costruzione della classe
 ClassA
a.doSomething(); // richiama il metodo doSomething() di ClassA
var b:ClassB = new ClassB(); // funzione di costruzione della classe
 ClassB
b.doSomething(); // richiama il metodo doSomething() di ClassB
```

Grazie alla tipizzazione degli oggetti in questo esempio di codice, il compilatore può impedire l'eventuale accesso a proprietà o metodi non definiti nella classe personalizzata.

Per ulteriori informazioni sulla tipizzazione forte dei dati, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344. Un'eccezione alla tipizzazione degli oggetti si verifica nel caso in cui la classe sia dichiarata dinamica per mezzo della parola chiave `dynamic`. Vedere “[Creazione di classi dinamiche](#)” a pagina 233.

3. Prima di continuare, salvare le modifiche al file Flash.

In questa sezione sono state fornite nozioni generali relative alla creazione e all'uso delle classi nei documenti Flash. Non dimenticare che è anche possibile creare istanze di classi incorporate o ActionScript di primo livello (vedere “[Informazioni sulle operazioni con le classi incorporate](#)” a pagina 273).

Per ulteriori informazioni sull'uso di questi file di classe in un file Flash, vedere “[Assegnazione di una classe a simboli in Flash](#)” a pagina 255.

## Assegnazione di una classe a simboli in Flash

Una classe può anche essere assegnata a simboli utilizzabili in un file Flash, ad esempio un oggetto clip filmato sullo stage.

**Per assegnare una classe a un simbolo di clip filmato:**

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Selezionare File > Salva con nome, assegnare al file il nome `Animal.as` e salvarlo sul disco rigido.

- 3.** Immettere il codice seguente nella finestra Script:

```
class Animal {
 public function Animal() {
 trace("Animal::funzione di costruzione");
 }
}
```

Questo codice ActionScript crea una nuova classe denominata Animal che dispone di un metodo della funzione di costruzione che traccia una stringa e la visualizza nel pannello Output.

- 4.** Salvare le modifiche apportate al file ActionScript.
- 5.** Selezionare File > Nuovo e quindi selezionare Documento Flash per creare un nuovo file FLA, quindi fare clic su OK.
- 6.** Selezionare File > Salva con nome, assegnare al file il nome **animal\_test.fla** e salvarlo nella stessa cartella del file Animal.as creato al punto 2.
- 7.** Selezionare Inserisci > Nuovo simbolo per aprire la finestra di dialogo Crea un nuovo simbolo.
- 8.** Specificare **animal** come nome di simbolo e selezionare il pulsante di scelta Clip filmato.
- 9.** Fare clic sul pulsante Avanzato nell'angolo inferiore destro della finestra di dialogo Crea un nuovo simbolo per visualizzare ulteriori opzioni.  
Il pulsante Avanzato è disponibile quando ci si trova modalità di base della finestra di dialogo Crea un nuovo simbolo.
- 10.** Selezionare la casella di controllo Esporta per ActionScript nella sezione Concatenamento per collegare in modo dinamico istanze di questo simbolo ai documenti Flash in fase di runtime.
- 11.** Immettere il valore **animal\_id** in Identificatore e immettere **Animal** in Classe ActionScript 2.0 (affinché corrisponda al nome di classe specificato al punto 3).
- 12.** Selezionare la casella di controllo Esporta nel primo fotogramma e fare clic su OK per applicare le modifiche e chiudere la finestra di dialogo.
- 13.** Salvare il documento Flash e selezionare Controllo > Prova filmato.

Nel pannello Output viene visualizzato il testo della funzione di costruzione della classe Animal.

**NOTA**

Per modificare le proprietà di concatenamento del clip filmato, è possibile fare clic con il pulsante destro del mouse sul simbolo nella libreria del documento e selezionare Proprietà o Concatenamento dal menu di scelta rapida.

# Compilazione ed esportazione di classi

Per impostazione predefinita, le classi utilizzate in un file SWF vengono inserite in un pacchetto ed esportate nel primo fotogramma del file SWF. È inoltre possibile specificare un fotogramma diverso per inserire le classi in un pacchetto ed esportarle. Può essere utile, ad esempio, se un file SWF utilizza numerose classi che richiedono molto tempo per essere scaricate, come nel caso di componenti. Se le classi vengono esportate nel primo fotogramma, l'utente deve attendere il caricamento di tutto il codice delle classi per poter visualizzare il fotogramma. Specificando invece un fotogramma successivo nella linea temporale, nei primi fotogrammi è possibile visualizzare in fase di caricamento una breve animazione, mentre il codice delle classi viene scaricato nei fotogrammi successivi.

## Per specificare il fotogramma di esportazione delle classi di un documento Flash:

1. Selezionare File > Nuovo, quindi Documento Flash. Salvare il nuovo documento come **exportClasses.fla**.
2. Rinominare il livello predefinito a **content**, trascinare un componente ProgressBar dal pannello Componenti sullo stage e denominarlo con il nome istanza **my\_pb**.
3. Creare un nuovo livello, trascinarlo sul livello del contenuto e rinominarlo **actions**.
4. Aggiungere il codice ActionScript seguente al fotogramma 1 del livello actions della linea temporale principale:  
`my_pb.indeterminate = true;`
5. Creare un nuovo fotogramma chiave sul fotogramma 2 del livello actions e aggiungere il seguente codice di ActionScript:

```
var classesFrame:Number = 10;
if (_framesloaded < classesFrame) {
 trace(this.getBytesLoaded() + " di " + this.getBytesTotal() + " byte
caricati");
 gotoAndPlay(1);
} else {
 gotoAndStop(classesFrame);
}
```

6. Creare un nuovo fotogramma chiave sul fotogramma 10 del livello actions e aggiungere il seguente codice di ActionScript:  
`stop();`
7. Creare un nuovo fotogramma chiave sul fotogramma 10 del livello content e trascinare più componenti sullo stage.

- 8.** Fare clic con il pulsante destro del mouse su ogni componente (tranne al componente ProgressBar) nel pannello Libreria e selezionare Concatenamento dal menu di scelta rapida per visualizzare la finestra di dialogo Proprietà del concatenamento.
- 9.** Nella finestra di dialogo Proprietà di concatenamento, accertarsi che Esporta per ActionScript sia selezionato, deselezionare la casella di controllo Esporta nel primo fotogramma e fare clic su OK.
- 10.** Scegliere File > Impostazioni pubblicazione.
- 11.** Nella finestra di dialogo Impostazioni pubblicazione, selezionare la scheda Flash.
- 12.** Per aprire la finestra di dialogo Impostazioni di ActionScript, fare clic sul pulsante Impostazioni visualizzato accanto al menu a comparsa per la scelta della versione di ActionScript.
- 13.** Nella casella di testo Esporta fotogramma per le classi, immettere il numero del fotogramma in cui si desidera esportare il codice delle classi (fotogramma 10).  
Se il fotogramma specificato non è presente nella linea temporale, al momento della pubblicazione del file SWF compare un messaggio di errore.
- 14.** Fare clic su OK per chiudere la finestra di dialogo Impostazioni di ActionScript, quindi fare clic su OK per chiudere la finestra di dialogo Impostazioni pubblicazione.
- 15.** Selezionare Controllo > Prova filmato per provare il documento Flash. Se i componenti vengono caricati troppo rapidamente, selezionare Visualizza > Simula scaricamento dal file SWF. Flash simula lo scaricamento del documento Flash a minore velocità, consentendo di visualizzare l'animazione del componente barra di avanzamento durante lo scarico dei file di classe.

Per ulteriori informazioni sui file ASO, vedere “[Utilizzo dei file ASO](#)” a pagina 258.

## Utilizzo dei file ASO

Durante la compilazione, Flash a volte crea file con estensione .aso nella sottodirectory /aso della directory del percorso di classe globale predefinito. Vedere “[Informazioni sull'impostazione e la modifica del percorso di classe](#)” a pagina 214. L'estensione .aso è l'abbreviazione di *ActionScript object* (oggetto ActionScript). Per ogni file ActionScript 2.0 importato in modo implicito o esplicito e compilato, Flash genera un file ASO che contiene il codice byte prodotto dal file ActionScript (AS) associato. Questi file contengono quindi la classe in formato compilato (il *codice byte*).

Flash deve rigenerare un file ASO solo quando si verifica la situazione seguente:

- Il file AS corrispondente è stato modificato.
- I file ActionScript che contengono definizioni importate o utilizzate dal file ActionScript corrispondente sono stati modificati.
- I file ActionScript inclusi dal file ActionScript corrispondente sono stati modificati.

Il compilatore crea file ASO per motivi di memorizzazione nella cache. La prima compilazione risulta più lenta rispetto alle successive perché solo i file AS modificati vengono ricompilati nei file ASO. Per i file AS invariati, il compilatore legge il codice byte già compilato direttamente dal file ASO e non ricompila il file AS.

Il formato di file ASO è un formato intermedio sviluppato solo per uso interno. Non è documentato e non ne è prevista la ridistribuzione.

Se si ritiene che Flash compili versioni precedenti di un file che è stato modificato, eliminare i file ASO ed eseguire nuovamente la compilazione. I file ASO devono essere eliminati quando Flash non esegue altre operazioni, ad esempio il controllo della sintassi o l'esportazione di file SWF.

### **Per eliminare i file ASO:**

Se si sta modificando un file FLA e si desidera eliminare un file ASO, selezionarne uno dei seguenti nell'ambiente di creazione:

- Selezionare Controllo > Elimina file ASO per eliminare i file ASO e proseguire la modifica.
- Selezionare Controllo > Elimina file ASO e prova filmato per eliminare i file ASO e provare l'applicazione.

Se si sta modificando un documento ActionScript nella finestra Script:

- Selezionare Controllo > Elimina file ASO per eliminare i file ASO e proseguire la modifica.
- Selezionare Controllo > Elimina file ASO e prova progetto per eliminare i file ASO e provare l'applicazione.

Esiste un limite alla quantità di codice che può essere inserita in una sola classe: il codice byte di una definizione di classe presente in un file SWF esportato non può superare i 32.767 byte. Se questo limite viene superato, appare un messaggio di avviso.

In genere le classi che contengono fino a 1500 righe di codice non superano questo limite.

Se il limite viene superato, spostare parte del codice in una classe diversa. È buona norma non scrivere codice di classe eccessivamente lungo.

# Nozioni fondamentali sulle classi e l'area di validità

Se si sposta codice ActionScript all'interno di classi, potrebbe essere necessario modificare il modo in cui viene utilizzata la parola chiave `this`. Se, ad esempio, è presente un metodo di classe che utilizza una funzione di callback (ad esempio il metodo `onLoad` della classe `LoadVars`), può risultare difficile comprendere se la parola chiave `this` si riferisce alla classe o all'oggetto `LoadVars`. In questo caso, potrebbe essere necessario creare un puntatore alla classe corrente, come illustrato nell'esempio seguente.

## Per comprendere l'area di validità e i file di classe esterni:

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
2. Immettere o incollare il codice seguente nella finestra Script:

```
/**
 * Classe Product
 * Product.as
 */
class Product {
 private var productsXml:XML;
 // Funzione di costruzione
 // targetXmlStr - stringa, contiene il percorso di un file XML
 function Product(targetXmlStr:String) {
 /* Crea un riferimento locale alla classe corrente.
 * Anche all'interno del gestore di eventi onLoad di XML
 * è possibile fare riferimento alla classe corrente anziché solo al
 * pacchetto XML.
 */
 var thisObj:Product = this;
 // Crea una variabile locale che viene utilizzata per caricare il
 // file XML.
 var prodXml:XML = new XML();
 prodXml.ignoreWhite = true;
 prodXml.onLoad = function(success:Boolean) {
 if (success) {
 /* Se il file XML viene caricato e analizzato correttamente,
 * impostare la variabile productsXml della classe sul documento
 * XML analizzato e chiamare la funzione init.
 */
 thisObj.productsXml = this;
 thisObj.init();
 } else {
 /* Si è verificato un errore durante il caricamento del file XML.
 */
 trace("errore durante il caricamento del file XML");
 }
 }
 }
}
```

```

 };
 // Inizia a caricare il documento XML.
 prodXml.load(targetXmlStr);
}
public function init():Void {
 // Visualizza il pacchetto XML.
 trace(this.productsXml);
}
}

```

Dato che si tenta di fare riferimento alla variabile di un membro privato all'interno di un gestore `onLoad`, la parola chiave `this` si riferisce all'istanza `prodXml` e non alla classe `Product`, come si potrebbe supporre. Per questo motivo, è necessario creare un puntatore al file della classe locale affinché sia possibile fare riferimento direttamente alla classe dal gestore `onLoad`. La classe può ora essere utilizzata con un documento Flash.

- 3.** Salvare il codice ActionScript precedente come **Product.as**.
- 4.** Creare un nuovo documento Flash denominato **testProduct.fla** nella stessa directory.
- 5.** Selezionare il fotogramma 1 della linea temporale.
- 6.** Immettere il codice ActionScript seguente nel pannello Azioni:

```
var myProduct:Product = new Product("http://www.helpexamples.com/
crossdomain.xml");
```

- 7.** Selezionare Controllo > Prova filmato per provare il codice nell'ambiente di prova.

Nel pannello Output viene visualizzato il contenuto del documento XML specificato.

Un ulteriore tipologia di area di validità che si può presentare durante il lavoro con queste classi è rappresentata dalle variabili statiche e dalle funzioni statiche. La parola chiave `static` specifica che una variabile o una funzione viene creata solo una volta per ogni classe, anziché in ogni istanza della classe. È possibile accedere a un membro di classe statico senza creare un'istanza della classe, utilizzando la sintassi `nomeClasse.nomeutente`. Per ulteriori informazioni sulle variabili e le funzioni statiche, vedere “[Informazioni su metodi e proprietà \(membri\) pubblici, privati e statici](#)” a pagina 220 e “[Uso dei membri di classe](#)” a pagina 228.

Un ulteriore vantaggio derivante dall'uso delle variabili statiche è il fatto che esse non perdono il loro valore al termine della relativa area di validità. L'esempio seguente dimostra l'uso della parola chiave `static` per creare un contatore che tenga traccia del numero di istanze della classe create da Flash. Dato che la variabile `numInstances` è statica, viene creata una sola volta per tutta la classe e non per ogni istanza della classe.

**Per utilizzare la parola chiave static:**

1. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.

2. Immettere il codice seguente nella finestra Script:

```
class User {
 private static var numInstances:Number = 0;
 public function User() {
 User.numInstances++;
 }
 public static function get instances():Number {
 return User.numInstances;
 }
}
```

Il codice riportato sopra definisce una classe User che tiene traccia del numero di chiamate alla funzione di costruzione. All'interno del metodo della funzione di costruzione viene incrementata una variabile statica privata (User.numInstances).

3. Salvare il documento come **User.as**.

4. Selezionare File > Nuovo e quindi Documento Flash per creare un nuovo file FLA e salvarlo nella stessa directory del file User.as.

5. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
trace(User.instances); // 0
var user1:User = new User();
trace(User.instances); // 1
var user2:User = new User();
trace(User.instances); // 2
```

La prima riga di codice chiama il metodo getter statico `instances()` che restituisce il valore della variabile statica privata `numInstances`. Il resto del codice crea nuove istanze della classe User e visualizza il valore corrente restituito dal metodo getter `instances()`.

6. Selezionare Controllo > Prova filmato per provare i documenti.

Per informazioni sull'uso della parola chiave `this` nelle classi, vedere “[Informazioni sull'uso della parola chiave this nelle classi](#)” a pagina 237.

# Informazioni sulle classi incorporate e di primo livello

Oltre agli elementi e ai costrutti di base del linguaggio ActionScript (ad esempio, i cicli `for` e `while`) e ai tipi di dati di base (numeri, stringhe e valori booleani) descritti in precedenza in questo manuale (vedere [Capitolo 10, “Dati e tipi di dati” a pagina 333](#) e [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi” a pagina 75](#)), ActionScript fornisce anche diverse classi incorporate (*tipi di dati complessi*) che offrono un'ampia varietà di opzioni e funzionalità per la creazione di script. Nei capitoli precedenti sono state utilizzate classi di primo livello e altre classi incorporate che fanno parte del linguaggio ActionScript e verranno utilizzate anche nei prossimi capitoli. Molte classi fornite con Flash possono essere utilizzate per aggiungere interattività e funzionalità nei file SWF e consentono anche di creare applicazioni complesse. La classe `Math`, ad esempio, può essere utilizzata per eseguire equazioni nelle applicazioni e la classe `BitmapData` per creare pixel e animazioni tramite script.

Le classi di primo livello, elencate in [“Classi di primo livello” a pagina 265](#), sono incorporate in Flash Player. Nella casella degli strumenti Azioni, tali classi si trovano nella directory Classes di ActionScript 2.0. Alcune si basano sulla specifica del linguaggio ECMAScript (ECMA-262) edizione 3 e sono dette *classi ActionScript di base*. A questo tipo di classi appartengono, ad esempio, `Array`, `Boolean`, `Date` e `Math`. Per ulteriori informazioni sui pacchetti, vedere [“Operazioni con i pacchetti” a pagina 202](#).

Le classi di ActionScript sono installate sul disco rigido. Le cartelle delle classi sono:

- Windows: Disco rigido\Documents and Settings\utente\Impostazioni locali\Dati applicazioni\Macromedia\Flash 8\lingua\Configuration\Classes.
- Macintosh: Disco rigido/Users/utente/Library/Supporto Applicazioni/Macromedia/Flash 8/lingua/Configuration/Classes.

Si esamini il documento Leggimi che si trova in questa directory per ulteriori informazioni sulla struttura.

Per comprendere la differenza tra le classi ActionScript di base e quelle specifiche di Flash, è possibile prendere come riferimento la distinzione tra il linguaggio JavaScript di base e client-side. Mentre le classi JavaScript client-side forniscono il controllo sull'ambiente client (il browser e il contenuto della pagina Web), le classi specifiche di Flash forniscono il controllo in fase di runtime sull'aspetto e sul comportamento di un'applicazione Flash.

Le restanti classi ActionScript incorporate sono specifiche di Macromedia Flash e del modello a oggetti di Flash Player e comprendono, ad esempio, le classi Camera, MovieClip e LoadVars. Altre classi sono organizzate in pacchetti, ad esempio flash.display. Tutte queste classi sono anche dette classi *incorporate* ovvero classi predefinite che possono essere utilizzate per aggiungere funzionalità alle applicazioni.

Tutte le sezioni seguenti introducono le classi ActionScript incorporate e descrivono le operazioni comuni che è possibile eseguire utilizzando queste classi incorporate. Per una panoramica sull'uso delle classi e degli oggetti nella programmazione orientata agli oggetti, vedere [“Informazioni sulle operazioni con le classi incorporate” a pagina 273](#). Esempi di codice che utilizzano queste classi sono inclusi in tutto il manuale *Apprendimento di ActionScript 2.0 in Flash*.

Per informazioni sugli elementi del linguaggio, ad esempio costanti, operatori e direttive, vedere [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi” a pagina 75](#).

Per ulteriori informazioni sulle operazioni con le classi di primo livello e incorporate, consultare i seguenti argomenti:

- [“Classi di primo livello” a pagina 265](#)
- [“Il pacchetto flash.display” a pagina 269](#)
- [“Il pacchetto flash.external” a pagina 269](#)
- [“Il pacchetto flash.filters” a pagina 270](#)
- [“Il pacchetto flash.geom” a pagina 271](#)
- [“Il pacchetto flash.net” a pagina 271](#)
- [“Il pacchetto flash.text” a pagina 272](#)
- [“Il pacchetto mx.lang” a pagina 272](#)
- [“I pacchetti System e TextField” a pagina 272](#)

## Altri elementi del linguaggio

Esistono altri elementi del linguaggio che costituiscono ActionScript, oltre alle classi. Essi comprendono direttive, costanti, funzioni globali, proprietà globali, operatori e istruzioni. Per informazioni su come utilizzare ciascuno di questi elementi del linguaggio, consultare i seguenti argomenti:

- [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi”](#)
- [Capitolo 5, “Funzioni e metodi”](#)

È possibile trovare un elenco di questi elementi del linguaggio nelle sezioni seguenti della *Guida di riferimento di ActionScript 2.0*:

- Compiler Directives
- Constants
- Global Functions
- Global Properties
- Operators
- Statements

## Classi di primo livello

Il primo livello contiene le classi e le funzioni globali ActionScript, molte delle quali forniscono funzionalità di base per le applicazioni. *Le classi di base*, basate sulla specifica ECMAScript, comprendono Array, Boolean, Date, Error, Function, Math, Number, Object, String e System. Per avere maggiori informazioni su ogni classe, vedere la tabella seguente.



Le classi CustomActions e XMLUI sono disponibili esclusivamente nell'ambiente di creazione di Flash.

| Classe        | Descrizione                                                                                                                                                                                                                                                                                                                  |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Accessibility | La classe Accessibility gestisce la comunicazione tra i file SWF e le applicazioni screen reader. I metodi di questa classe vengono utilizzati con la proprietà globale <code>_accProps</code> per controllare le proprietà accessibili di clip filmato, pulsanti e campi di testo in fase di runtime. Vedere Accessibility. |
| Array         | La classe Array rappresenta gli array di ActionScript; tutti gli oggetti array sono istanze di questa classe. La classe Array fornisce i metodi e le proprietà per eseguire operazioni con oggetti Array. Vedere Array.                                                                                                      |
| AsBroadcaster | Fornisce funzionalità di notifica eventi e gestione listener che possono essere aggiunte ad altri oggetti. Vedere AsBroadcaster.                                                                                                                                                                                             |
| Boolean       | La classe Boolean è un wrapper per i valori booleani ( <code>true</code> o <code>false</code> ). Vedere Boolean.».                                                                                                                                                                                                           |
| Button        | La classe Button fornisce metodi, proprietà e gestori di eventi per eseguire le operazioni con i pulsanti. Vedere Button. Si noti che la classe incorporata Button è diversa dalla classe di componenti Button, associata al componente della versione 2, Button.                                                            |

| <b>Classe</b>   | <b>Descrizione</b>                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Camera          | La classe Camera fornisce accesso alla videocamera dell'utente, se installata. Se utilizzato con Flash Communication Server, il file SWF può acquisire, trasmettere e registrare immagini e video provenienti da una videocamera. Vedere Camera.                                                                                                                                                                                              |
| Color           | La classe Color consente di impostare il valore RGB del colore e il valore di trasformazione del colore di istanze di clip filmato e di recuperare tali valori dopo la loro impostazione. La classe Color è sconsigliata in Flash Player 8 a favore della classe ColorTransform. Per informazioni sul valore di trasformazione del colore, vedere ColorTransform (flash.geom.ColorTransform).                                                 |
| ContextMenu     | La classe ContextMenu consente di controllare il contenuto del menu di scelta rapida di Flash Player in fase di runtime. È possibile associare oggetti ContextMenu separati agli oggetti MovieClip, Button o TextField utilizzando la proprietà <code>menu</code> disponibile per tali classi. È inoltre possibile aggiungere voci di menu personalizzate a un oggetto ContextMenu utilizzando la classe ContextMenuItem. Vedere ContextMenu. |
| ContextMenuItem | La classe ContextMenuItem consente di creare nuove voci di menu da visualizzare nel menu di scelta rapida di Flash Player. Utilizzando la classe ContextMenu, è possibile aggiungere le nuove voci di menu create con la classe ContextMenuItem al menu di scelta rapida di Flash Player. Vedere ContextMenuItem.                                                                                                                             |
| CustomActions   | La classe CustomActions consente di gestire qualsiasi azione personalizzata registrata mediante lo strumento di creazione. Vedere CustomActions.                                                                                                                                                                                                                                                                                              |
| Date            | La classe Date determina il tipo di rappresentazione di data e ora in ActionScript e supporta operazioni di manipolazione di data e ora. Consente inoltre di ottenere la data e l'ora correnti dal sistema operativo. Vedere Date.                                                                                                                                                                                                            |
| Error           | La classe Error contiene informazioni sugli errori di runtime che si verificano negli script. Per generare una condizione di errore viene generalmente utilizzata l'istruzione <code>throw</code> che è possibile poi gestire tramite l'istruzione <code>try..catch..finally</code> . Vedere Error.                                                                                                                                           |
| Function        | La classe Function è la rappresentazione di classe di tutte le funzioni ActionScript, comprese quelle native di ActionScript e quelle definite dall'utente. Vedere Function.                                                                                                                                                                                                                                                                  |
| Key             | La classe Key fornisce i metodi e le proprietà per accedere alle informazioni sulla tastiera e i tasti premuti. Vedere Key.                                                                                                                                                                                                                                                                                                                   |

---

| <b>Classe</b>   | <b>Descrizione</b>                                                                                                                                                                                                                                                                                                                              |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LoadVars        | La classe LoadVars consente di trasferire variabili tra un file SWF e un server in coppie nome/valore. Vedere LoadVars.                                                                                                                                                                                                                         |
| LocalConnection | La classe LocalConnection consente di sviluppare file SWF in grado di inviare istruzioni gli uni agli altri senza utilizzare il metodo <code>fscommand()</code> di JavaScript. Vedere LocalConnection.                                                                                                                                          |
| Math            | La classe Math consente di accedere in modo semplice alle più comuni costanti matematiche e comprende diverse funzioni matematiche standard. Tutte le proprietà e i metodi della classe Math sono statici e devono essere chiamati con l'ausilio della sintassi <code>Math.metodo(parametro)</code> o <code>Math.costante</code> . Vedere Math. |
| Microphone      | La classe Microphone fornisce accesso al microfono dell'utente, se installato. Se utilizzato con Flash Communication Server, il file SWF può trasmettere e registrare l'audio proveniente da un microfono. Vedere Microphone.                                                                                                                   |
| Mouse           | La classe Mouse consente di controllare il mouse in un file SWF; tale classe consente, ad esempio, di mostrare o nascondere il puntatore. Vedere Mouse.                                                                                                                                                                                         |
| MovieClip       | Ogni clip filmato in un file SWF è un'istanza della classe MovieClip. È possibile utilizzare i metodi e le proprietà di questa classe per controllare gli oggetti del clip filmato. Vedere MovieClip.                                                                                                                                           |
| MovieClipLoader | Questa classe consente di implementare funzioni di callback del listener che forniscono le informazioni sullo stato mentre i file SWF, JPEG, GIF e PNG vengono caricati nelle istanze dei clip filmato. Vedere MovieClipLoader.                                                                                                                 |
| NetConnection   | La classe NetConnection stabilisce una connessione in streaming locale per la riproduzione di un file Flash Video (FLV) da un indirizzo HTTP o dal file system locale. Vedere NetConnection.                                                                                                                                                    |
| NetStream       | La classe NetStream controlla la riproduzione dei file FLV da un file system locale o un indirizzo HTTP. Vedere NetStream.                                                                                                                                                                                                                      |
| Number          | La classe Number è un wrapper per il tipo di dati numerico di base. Vedere Number.                                                                                                                                                                                                                                                              |
| Object          | La classe Object si trova alla radice della gerarchia di classi ActionScript; tutte le altre classi ereditano i suoi metodi e proprietà. Vedere Object.                                                                                                                                                                                         |
| PrintJob        | La classe PrintJob consente di stampare il contenuto di un file SWF, compreso il contenuto di cui è stato eseguito il rendering dinamico, e di documenti su più pagine. Vedere PrintJob.                                                                                                                                                        |

---

| <b>Classe</b> | <b>Descrizione</b>                                                                                                                                                                                                                                                                                                                                           |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Selection     | La classe Selection consente di impostare e controllare il campo di testo in cui si trova il punto di inserimento, ovvero il campo attivo. Vedere Selection.                                                                                                                                                                                                 |
| SharedObject  | La classe SharedObject garantisce la memorizzazione dei dati locali persistenti sul computer client, analogamente ai cookie, e la condivisione dati in tempo reale tra gli oggetti sul computer client. Vedere SharedObject.                                                                                                                                 |
| Sound         | La classe Sound consente di controllare l'audio di un file SWF. Vedere Sound.                                                                                                                                                                                                                                                                                |
| Stage         | La classe Stage fornisce informazioni sulle dimensioni, l'allineamento e la modalità di ridimensionamento di un file SWF e comunica gli eventi di ridimensionamento sullo stage. Vedere Stage.                                                                                                                                                               |
| String        | La classe String è un wrapper per il tipo di dati di base stringa che consente di utilizzare i metodi e le proprietà dell'oggetto String per manipolare i tipi di valore stringa di base. Vedere String.                                                                                                                                                     |
| System        | La classe System fornisce informazioni su Flash Player e sul sistema in cui Flash Player viene eseguito (ad esempio la risoluzione dello schermo e la lingua del sistema attualmente impostata). Consente inoltre di mostrare o nascondere il pannello Impostazioni di Flash Player e modificare le impostazioni di sicurezza per i file SWF. Vedere System. |
| TextField     | La classe TextField fornisce il controllo su campi di testo di input e dinamici, consentendo ad esempio il recupero di informazioni, il richiamo di gestori di eventi e la modifica di proprietà come alfa o il colore di sfondo. Vedere TextField.                                                                                                          |
| TextFormat    | La classe TextFormat consente di applicare gli stili di formattazione ai caratteri o paragrafi di un oggetto TextField. Vedere TextFormat.                                                                                                                                                                                                                   |
| TextSnapshot  | L'oggetto TextSnapshot consente di accedere a testo statico ed eseguirne il layout all'interno di un clip filmato. Vedere TextSnapshot.                                                                                                                                                                                                                      |
| Video         | La classe Video consente di visualizzare oggetti video in un file SWF. Può essere utilizzata con Flash Communication Server per visualizzare streaming video dal vivo in un file SWF o all'interno di Flash per visualizzare un file Flash Video (FLV). Vedere Video.                                                                                        |
| XML           | Questa classe fornisce i metodi e le proprietà per eseguire operazioni con oggetti XML. Vedere XML.                                                                                                                                                                                                                                                          |
| XMLNode       | La classe XMLNode rappresenta un singolo nodo in un albero di documento XML. È la superclasse della classe XML. Vedere XMLNode.                                                                                                                                                                                                                              |

| <b>Classe</b> | <b>Descrizione</b>                                                                                                                                                                                                                                                                                                                |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XMLSocket     | La classe XMLSocket consente di creare una connessione socket permanente tra un computer server e un client su cui è in esecuzione Flash Player. I socket client consentono il trasferimento di dati a bassa latenza, come richiesto, ad esempio, per le applicazioni di conversazione in linea in tempo reale. Vedere XMLSocket. |
| XMLUI         | L'oggetto XMLUI consente la comunicazione con i file SWF utilizzati come interfaccia utente personalizzata per le funzioni di estensibilità dello strumento di creazione di Flash (ad esempio Comportamenti, Comandi, Effetti e Strumenti). Vedere XMLUI.                                                                         |

## Il pacchetto flash.display

Il pacchetto flash.display contiene la classe BitmapData utilizzabile per creare contenuto grafico.

| <b>Classe</b> | <b>Descrizione</b>                                                                                                                                                                                                       |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BitmapData    | La classe BitmapData consente di creare nel documento immagini bitmap opache o trasparenti, di dimensioni arbitrarie, e di manipolarle in diversi modi in fase di runtime. Vedere BitmapData (flash.display.BitmapData). |

## Il pacchetto flash.external

Il pacchetto flash.external consente di comunicare con il contenitore Flash Player tramite codice ActionScript. Se, ad esempio, si incorpora un file SWF in una pagina HTML, la pagina è il contenitore. È possibile comunicare con la pagina HTML tramite la classe ExternalInterface e JavaScript. Chiamata anche API esterna.

| <b>Classe</b>     | <b>Descrizione</b>                                                                                                                                                                                                                                                                                               |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ExternalInterface | La classe ExternalInterface è l'API External, un sottosistema che consente la comunicazione tra ActionScript e il contenitore Flash Player, ad esempio una pagina HTML che utilizza JavaScript o un'applicazione desktop che utilizza Flash Player. Vedere ExternalInterface (flash.external.ExternalInterface). |

## Il pacchetto flash.filters

Il pacchetto flash.filters contiene classi per gli effetti di filtraggio delle immagini bitmap disponibili in Flash Player 8. I filtri consentono di applicare diversi effetti visivi, quale la sfocatura, la smussatura e le ombre esterne alle istanze di Image e MovieClip. Per ulteriori informazioni su ogni classe, vedere i riferimenti incrociati contenuti nella tabella seguente.

| Classe                | Descrizione                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BevelFilter           | <i>La classe BevelFilter consente di aggiungere un effetto di smussatura a un'istanza di clip filmato. Vedere BevelFilter (flash.filters.BevelFilter).</i>                                                                                                                                                                                         |
| BitmapFilter          | <i>La classe BitmapFilter è una classe base per tutti gli effetti filtro. Vedere BitmapFilter (flash.filters.BitmapFilter).</i>                                                                                                                                                                                                                    |
| BlurFilter            | <i>La classe BlurFilter consente di applicare un effetto di sfocatura a istanze di clip filmato. Vedere BlurFilter (flash.filters.BlurFilter).</i>                                                                                                                                                                                                 |
| ColorMatrixFilter     | <i>La classe ColorMatrixFilter consente di applicare una trasformazione matrice 4x5 al colore ARGB e ai valori alfa di ogni pixel dell'immagine di input. Dopo avere applicato la trasformazione, è possibile produrre un risultato con un nuovo set di colori ARGB e valori alfa. Vedere ColorMatrixFilter (flash.filters.ColorMatrixFilter).</i> |
| ConvolutionFilter     | <i>La classe ConvolutionFilter consente di applicare un effetto filtro di convoluzione matrice. Vedere ConvolutionFilter (flash.filters.ConvolutionFilter).</i>                                                                                                                                                                                    |
| DisplacementMapFilter | <i>La classe DisplacementMapFilter consente di utilizzare i valori di pixel di un'immagine specificata (l'immagine della mappa di spostamento) per spostare l'istanza originale (un clip filmato) a cui viene applicato un filtro. Vedere DisplacementMapFilter (flash.filters.DisplacementMapFilter).</i>                                         |
| DropShadowFilter      | <i>La classe DropShadowFilter consente di aggiungere un'ombra esterna a un clip filmato. Vedere DropShadowFilter (flash.filters.DropShadowFilter).</i>                                                                                                                                                                                             |
| GlowFilter            | <i>La classe GlowFilter consente di aggiungere un effetto di bagliore a un clip filmato. Vedere GlowFilter (flash.filters.GlowFilter).</i>                                                                                                                                                                                                         |
| GradientBevelFilter   | <i>La classe GradientBevelFilter consente di applicare un effetto di smussatura con gradiente a un clip filmato. Vedere GradientBevelFilter (flash.filters.GradientBevelFilter).</i>                                                                                                                                                               |
| GradientGlowFilter    | <i>La classe GradientGlowFilter consente di applicare un effetto di bagliore con gradiente a un clip filmato. Vedere GradientGlowFilter (flash.filters.GradientGlowFilter).</i>                                                                                                                                                                    |

## Il pacchetto flash.geom

Il pacchetto flash.geom contiene classi per oggetti geometrici, ad esempio punti, rettangoli e matrici di trasformazione, che supportano la classe `BitmapData` e la funzione di caching delle bitmap. Per ulteriori informazioni su ogni classe, vedere i riferimenti incrociati contenuti nella tabella seguente.

| Classe                      | Descrizione                                                                                                                                                                                                                                                                                        |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ColorTransform</code> | La classe <code>ColorTransform</code> consente di impostare matematicamente il valore dei colori RGB e la trasformazione dei colori per un'istanza. I valori possono essere recuperati dopo essere stati impostati. Vedere <code>ColorTransform</code> ( <code>flash.geom.ColorTransform</code> ). |
| <code>Matrix</code>         | Rappresenta una matrice di trasformazione che determina come mappare punti da uno spazio di coordinate a un altro. Vedere <code>Matrix</code> ( <code>flash.geom.Matrix</code> ).                                                                                                                  |
| <code>Point</code>          | L'oggetto <code>Point</code> rappresenta una posizione in un sistema di coordinate a due dimensioni, dove <code>x</code> rappresenta l'asse orizzontale e <code>y</code> rappresenta l'asse verticale. Vedere <code>Point</code> ( <code>flash.geom.Point</code> ).                                |
| <code>Rectangle</code>      | La classe <code>Rectangle</code> viene utilizzata per creare e modificare oggetti <code>Rectangle</code> . Vedere <code>Rectangle</code> ( <code>flash.geom.Rectangle</code> ).                                                                                                                    |
| <code>Transform</code>      | Raccoglie i dati sulle trasformazioni di colore e le modifiche di coordinate applicate a un'istanza di un oggetto. Vedere <code>Transform</code> ( <code>flash.geom.Transform</code> ).                                                                                                            |

## Il pacchetto flash.net

Il pacchetto flash.net contiene classi che consentono di caricare e scaricare uno o più file tra il computer di un utente e il server. Per ulteriori informazioni su ogni classe, vedere i riferimenti incrociati contenuti nella tabella seguente.

| Classe                         | Descrizione                                                                                                                                                                                                         |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>FileReference</code>     | La classe <code>FileReference</code> consente di caricare e scaricare uno o più file tra il computer di un utente e un server. Vedere <code>FileReference</code> ( <code>flash.net.FileReference</code> ).          |
| <code>FileReferenceList</code> | La classe <code>FileReferenceList</code> consente di caricare e scaricare uno o più file dal computer di un utente a un server. Vedere <code>FileReferenceList</code> ( <code>flash.net.FileReferenceList</code> ). |

## Il pacchetto flash.text

Il pacchetto flash.text contiene la classe TextRenderer per l'antialiasing avanzato disponibile in Flash Player 8.

| Classe       | Descrizione                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------|
| TextRenderer | Questa classe comprende funzionalità per l'antialiasing avanzato in Flash Player 8. Vedere TextRenderer (flash.text.TextRenderer). |

## Il pacchetto mx.lang

Il pacchetto mx.lang contiene la classe Locale per i testi in più lingue.

| Classe | Descrizione                                                                                                                      |
|--------|----------------------------------------------------------------------------------------------------------------------------------|
| Locale | Questa classe permette di controllare la visualizzazione del testo in più lingue in un file SWF. Vedere Locale (mx.lang.Locale). |

## I pacchetti System e TextField

Il pacchetto System contiene le classi capabilities, IME e security, che consentono di modificare le impostazioni client che possono incidere sul funzionamento dell'applicazione in Flash Player. Per ulteriori informazioni sulle singole classi, vedere i riferimenti incrociati contenuti nella tabella seguente.

| Classe       | Descrizione                                                                                                                                                                                                            |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| capabilities | La classe Capabilities determina la capacità del sistema e di Flash Player in cui viene caricato il file SWF e consente di personalizzare il contenuto per formati diversi. Vedere capabilities (System.capabilities). |
| IME          | La classe IME consente di manipolare direttamente l'IME (Input Method Editor) del sistema operativo all'interno dell'applicazione Flash Player in esecuzione su un computer client. Vedere IME (System.IME).           |
| security     | La classe Security contiene i metodi che specificano il modo in cui file SWF di diversi domini possono comunicare gli uni con gli altri. Vedere security (System.security).                                            |

Il pacchetto TextField contiene la classe StyleSheet che consente di applicare stili CSS al testo.

| Classe     | Descrizione                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StyleSheet | La classe StyleSheet consente di creare un oggetto foglio di stile contenente regole di formattazione del testo, quali la dimensione del carattere, il colore e altri stili di formattazione. Vedere StyleSheet (TextField.StyleSheet). |

## Informazioni sulle operazioni con le classi incorporate

Nella programmazione orientata agli oggetti (OOP), una *classe* definisce una categoria di oggetto. Una classe descrive le proprietà (dati) e il comportamento (metodi) di un oggetto in modo molto simile a come un progetto di architettura descrive le caratteristiche di un edificio. Per informazioni sulle classi e altri concetti della programmazione orientata agli oggetti, consultare le sezioni seguenti:

- “Nozioni fondamentali sulla programmazione orientata agli oggetti.” a pagina 204
- “Creazione di file di classi personalizzate” a pagina 208

Flash 8 dispone di numerose classi incorporate utilizzabili nel codice (vedere “[Informazioni sulle classi incorporate e di primo livello](#)” a pagina 263), che permettono di aggiungere facilmente funzioni di interattività alle applicazioni. Per utilizzare le proprietà e i metodi definiti da una classe incorporata, è necessario creare prima un’istanza di tale classe (a eccezione delle classi che non dispongono di membri statici). La relazione tra un’istanza e la relativa classe è simile a quella che intercorre tra un edificio e il relativo progetto architettonico, come spiegato in “[Informazioni sulle classi incorporate e di primo livello](#)” a pagina 263.

Per ulteriori informazioni sull’uso delle classi incorporate in Flash 8, consultare i seguenti argomenti:

- “Creazione di una nuova istanza di una classe incorporata” a pagina 274
- “Accesso alle proprietà di un oggetto incorporato” a pagina 274
- “Informazioni sulla chiamata dei metodi di un oggetto incorporato” a pagina 275
- “Informazioni sui membri delle classi (statici)” a pagina 275
- “Precaricamento dei file di classe” a pagina 277
- “Esclusione di classi” a pagina 276

## Creazione di una nuova istanza di una classe incorporata

Per creare un'istanza di una classe ActionScript, utilizzare l'operatore `new` per richiamare la funzione di costruzione della classe. La funzione di costruzione ha sempre lo stesso nome della classe e restituisce un'istanza della classe che generalmente viene assegnata a una variabile.

Il codice seguente, ad esempio, crea un nuovo oggetto Sound:

```
var song_sound:Sound = new Sound();
```

In alcuni casi non occorre creare un'istanza di una classe per utilizzarne i metodi e le proprietà. Per ulteriori informazioni, vedere “[Informazioni sui membri delle classi \(statici\)](#)” [a pagina 275](#).

## Accesso alle proprietà di un oggetto incorporato

È possibile usare l'operatore punto `(.)` per accedere al valore di una proprietà di un oggetto. Specificare il nome dell'oggetto a sinistra del punto e il nome della proprietà a destra. Ad esempio, nell'istruzione seguente `my_obj` è l'oggetto mentre `firstName` è la proprietà:

```
my_obj.firstName
```

Il codice seguente crea un nuovo oggetto Array, quindi ne visualizza la proprietà `length`.

```
var my_array:Array = new Array("apples", "oranges", "bananas");
trace(my_array.length); // 3
```

È inoltre possibile accedere alle proprietà di un oggetto, ad esempio a scopo di debug, utilizzando l'operatore di accesso agli array `([])`. Nell'esempio seguente è presente un ciclo relativo a un oggetto per visualizzare ciascuna delle proprietà dell'oggetto.

### Per eseguire un'elaborazione ciclica sul contenuto di un oggetto:

1. Creare un nuovo documento Flash e salvarlo come **forin.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var results:Object = {firstName:"Tommy", lastName:"G", age:7, avg:0.336,
 b:"R", t:"L"};
for (var i:String in results) {
 trace("the value of [" + i + "] is: " + results[i]);
}
```

Il codice riportato in precedenza definisce un nuovo Object denominato `results` e definisce i valori di `firstName`, `lastName`, `age`, `avg`, `b` e `t`. Un ciclo `for..in` traccia ogni proprietà dell'oggetto `results` e ne traccia il valore sul pannello Output.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Per ulteriori informazioni sugli operatori, compresi gli operatori punto e di accesso matrice, vedere “[Informazioni sugli operatori](#)” a pagina 143. Per ulteriori informazioni su metodi e proprietà, consultare il [Capitolo 5, “Funzioni e metodi”](#) a pagina 169. Per alcuni esempi sull'utilizzo delle proprietà della classe incorporata MovieClip, vedere [Capitolo 11, “Operazioni con i clip filmato”](#) a pagina 381. Per alcuni esempi sull'utilizzo delle proprietà delle classi TextField, String, TextRenderer e TextFormat, vedere [Capitolo 12, “Operazioni con il testo e le stringhe”](#) a pagina 415.

## Informazioni sulla chiamata dei metodi di un oggetto incorporato

Per chiamare un metodo di un oggetto si utilizza l'operatore punto (.) seguito dal metodo. Il codice seguente, ad esempio, crea un nuovo oggetto Sound e ne chiama il metodo setVolume():

```
var my_sound:Sound = new Sound(this);
my_sound.setVolume(50);
```

Per alcuni esempi sull'utilizzo dei metodi della classe incorporata MovieClip, vedere [Capitolo 11, “Operazioni con i clip filmato”](#) a pagina 381. Per alcuni esempi sull'utilizzo dei metodi delle classi incorporate TextField, String, TextRenderer e TextFormat, vedere [Capitolo 12, “Operazioni con il testo e le stringhe”](#) a pagina 415.

## Informazioni sui membri delle classi (statici)

Alcune classi ActionScript incorporate dispongono di *membri delle classi* (o *membri statici*). I membri delle classi (proprietà e metodi) vengono richiamati non sull'istanza della classe, ma sul nome della classe stessa ed è possibile accedere ad essi nello stesso modo. Non è pertanto necessario creare un'istanza della classe per utilizzarne metodi e proprietà.

Ad esempio, tutte le proprietà della classe Math sono statiche. Il codice seguente richiama il metodo max() della classe Math per determinare quale tra due numeri è il maggiore:

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Per ulteriori informazioni sui metodi statici della classe Math e alcuni esempi, vedere Math nella *Guida di riferimento di ActionScript 2.0*.

## Esclusione di classi

Per ridurre le dimensioni di un file SWF, è possibile escludere determinate classi dalla compilazione. Queste classi saranno comunque accessibili e disponibili per il controllo dei tipi. Questa procedura è consigliabile, ad esempio, se si sviluppa un'applicazione che utilizza più file SWF o librerie condivise, in particolare se accedono a molte classi uguali. L'esclusione di classi permette di impedirne la duplicazione all'interno dei file.

Per ulteriori informazioni sull'esclusione di classi, consultare i seguenti argomenti:

- “[Precaricamento dei file di classe](#)” a pagina 277

### Per escludere le classi dalla compilazione:

1. Creare un nuovo file XML.
2. Assegnare al file XML il nome *FLA\_nofile\_exclude.xml*, dove *FLA\_nofile* corrisponde al nome del file FLA senza estensione.  
Se il file FLA, ad esempio, è sellStocks.fla, il nome del file XML deve essere sellStocks\_exclude.xml.
3. Salvare il file nella stessa directory del file FLA.
4. Inserire i tag seguenti nel file XML:

```
<excludeAssets>
 <asset name="className1" />
 <asset name="className2" />
</excludeAssets>
```

I valori specificati per gli attributi name nei tag `<asset>` corrispondono ai nomi delle classi che si desidera escludere dal file SWF. Aggiungere tutti i valori necessari per l'applicazione. Con il file XML seguente, ad esempio, vengono escluse dal file SWF le classi `mx.core.UIObject` e `mx.screens.Slide`:

```
<excludeAssets>
 <asset name="mx.core.UIObject" />
 <asset name="mx.screens.Slide" />
</excludeAssets>
```

Per informazioni su come precaricare le classi, vedere “[Precaricamento dei file di classe](#)” a pagina 277.

## Precaricamento dei file di classe

In questa sezione sono descritte alcune delle metodologie adottabili per precaricare ed esportare componenti e classi in Flash 8 (incluse le classi usate dai componenti basati sulla versione 2 di Macromedia Component Architecture). Il *precaricamento* riguarda il caricamento di alcuni dati per un file SWF prima che l'utente inizi a interagire con il file. In caso di utilizzo di classi esterne, Flash importa le classi nel primo fotogramma di un file SWF. Questi dati sono il primo elemento che viene caricato in un file SWF. Un comportamento simile è adottato per le classi dei componenti, perché anche la struttura dei componenti viene caricata nel primo fotogramma di un file SWF. Se si creano applicazioni di grandi dimensioni, il tempo di caricamento può essere considerevole se si devono importare dati. Per questo motivo è necessario gestire i dati in modo intelligente, come illustrato di seguito.

Dato che le classi sono le prime informazioni che vengono caricate, si potrebbero verificare problemi con la creazione di una barra di avanzamento o con il caricamento di animazione, se si desidera che la barra rifletta l'avanzamento del caricamento di tutti i dati, comprese le classi, perché queste ultime vengono caricate prima della barra. Le classi dovranno quindi essere caricate dopo altre parti del file SWF, ma prima di utilizzare i componenti.

La procedura seguente illustra come modificare il fotogramma per il caricamento delle classi in un file SWF.

### **Per selezionare un fotogramma diverso per il caricamento delle classi in un file SWF:**

1. Scegliere File > Impostazioni pubblicazione.
2. Selezionare la scheda Flash e fare clic sul pulsante Impostazioni.
3. Nella casella di testo Esporta fotogramma per le classi, digitare il numero di un nuovo fotogramma per determinare quando caricare le classi.
4. Fare clic su OK.

Non sarà possibile utilizzare le classi fino a quando l'indicatore di riproduzione non raggiungerà il fotogramma in cui le classi devono essere caricate. Ad esempio, sarà necessario caricare i componenti dopo il fotogramma in cui verranno caricate le classi ActionScript 2.0 perché i componenti della versione 2 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 classi, come le classi di componenti della versione 2, è necessario precaricare i componenti nel file SWF. A questo scopo, impostare i componenti affinché siano associati a un fotogramma diverso nel file SWF. Per impostazione predefinita, i componenti dell'interfaccia utente vengono esportati nel fotogramma 1 del file SWF: è quindi necessario esser certi di avere deselezionato Esporta nel primo fotogramma nella finestra di dialogo Concatenamento del componente.

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 nella casella di controllo Esporta fotogramma per le classi.

**ATTENZIONE**

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

# Ereditarietà

Nel [Capitolo 6, “Classi”](#) è stata introdotta la creazione di file di classe ed è stato illustrato come le classi possono aiutare a organizzare il codice in file esterni. È stata inoltre dimostrata l'organizzazione dei file di classe in pacchetti correlati. Questo capitolo intende dimostrare come scrivere classi più avanzate che estendano le funzionalità di una classe esistente. È infatti possibile estendere classi personalizzate o esistenti per aggiungere nuovi metodi e proprietà.

Per ulteriori informazioni sull'ereditarietà, consultare [“Informazioni sull'ereditarietà” a pagina 279](#). Per ulteriori informazioni su metodi e proprietà, consultare il [Capitolo 5, “Funzioni e metodi” a pagina 169](#).

Per ulteriori informazioni su ereditarietà, consultare i seguenti argomenti:

<a href="#">Informazioni sull'ereditarietà</a> .....	279
<a href="#">Informazioni sulla creazione di sottoclassi in Flash</a> .....	281
<a href="#">Uso del polimorfismo in un'applicazione</a> .....	287

## Informazioni sull'ereditarietà

Nel [Capitolo 6, “Classi”](#) è stata illustrata la creazione di un file di classe allo scopo di definire tipi di dati personalizzati. Se si creano file di classe personalizzati, il codice non è più contenuto nella linea temporale, ma in file esterni e la sua modifica risulta più semplice. In questo capitolo viene presentata una tecnica della programmazione orientata agli oggetti detta *creazione di sottoclassi* o *estensione di una classe* che consente di creare nuove classi in base a una classe esistente.

Uno dei vantaggi assicurati dalla programmazione orientata agli oggetti è la possibilità di creare *sottoclassi* di una classe. La sottoclasse eredita tutte le proprietà e i metodi di una *superclasse*. Se, ad esempio, si estende la classe MovieClip (o se ne crea una *sottoclasse*), si crea una classe personalizzata che estende la classe MovieClip. La sottoclasse eredita tutte le proprietà e i metodi della classe MovieClip. In alternativa, è possibile creare un set di classi che estenda una superclasse personalizzata. La classe Pomodoro, ad esempio, potrebbe essere creata estendendo la superclasse Verdura.

La sottoclasse definisce in genere metodi e proprietà aggiuntivi utilizzabili nell'applicazione, ovvero *estende* la superclasse. Le sottoclassi possono inoltre sostituire i metodi ereditati da una superclasse, fornendone definizioni proprie. Se una sottoclasse sostituisce un metodo ereditato dalla relativa superclasse, non è più possibile accedere alla definizione della superclasse all'interno della stessa. L'unica eccezione alla regola precedente dipende dal fatto che, se si è all'interno della funzione di costruzione della classe, si chiama la costruzione della superclasse tramite l'istruzione `super`. Per ulteriori informazioni sulla sostituzione, vedere “[Sostituzione di metodi e proprietà](#)” a pagina 285.

È possibile, ad esempio, creare una classe `Mammifero` che definisca alcune proprietà e alcuni comportamenti comuni a tutti i mammiferi e, successivamente, creare una sottoclasse `Gatto` che estenda la classe `Mammifero`. L'uso di sottoclassi consente di riutilizzare il codice, in quanto estendendo una classe esistente si evita di ricreare tutto il codice comune a entrambe le classi. Un'ulteriore sottoclasse, ad esempio la classe `Siamese`, potrebbe a sua volta estendere la classe `Gatto`, e così via. In un'applicazione complessa, la definizione della struttura gerarchica delle classi rappresenta una buona parte del processo di progettazione.

L'ereditarietà e la creazione di sottoclassi possono rivelarsi molto utili in applicazioni di grandi dimensioni, perché permettono di creare una serie di classi correlate con funzionalità condivise. È possibile, ad esempio, creare una classe `Dipendente` che definisca i metodi e le proprietà di base di un dipendente tipico di una società e una classe `Collaboratore` che estenda la classe `Dipendente` e ne erediti tutti i metodi e le proprietà. La classe `Collaboratore` potrebbe aggiungere metodi e proprietà specifici o sostituire i metodi e le proprietà definiti nella superclasse `Dipendente`. È quindi possibile creare una nuova classe `Manager` che estenda a sua volta la classe `Dipendente` e definisca metodi e proprietà aggiuntivi come `assumere()`, `licenziare()`, `aumentare()` e `promuovere()`. È possibile anche estendere una sottoclasse, come `Manager`, e creare una nuova classe `Direttore` che aggiunga nuovi metodi o sostituisca i metodi esistenti.

Quando si estende una classe esistente, la nuova classe eredita tutti i metodi e le proprietà correnti della superclasse. Se le classi non fossero correlate, sarebbe necessario ricreare ogni metodo e proprietà in ogni file di classe, anche se le funzionalità sono le stesse nelle classi simili, perdendo tempo non solo nella scrittura del codice, ma anche nel debug dell'applicazione e nella gestione del progetto, nel caso in cui la logica simile debba essere modificata in più file.

In ActionScript, viene utilizzata la parola chiave `extends` per determinare l'ereditarietà tra una classe e la relativa superclasse o per estendere un'interfaccia. Per ulteriori informazioni sull'uso della parola chiave `extends`, consultare “[Informazioni sulla creazione di sottoclassi in Flash](#)” a pagina 281 e “[Informazioni sulla creazione di una sottoclasse](#)” a pagina 281. Per ulteriori informazioni sulla parola chiave `extends`, vedere `extends` statement nella *Guida di riferimento di ActionScript 2.0*.

# Informazioni sulla creazione di sottoclassi in Flash

Nella programmazione orientata agli oggetti, una sottoclasse può ereditare le proprietà e i metodi di un'altra classe, detta *superclasse*. È possibile estendere classi personalizzate e molte delle classi di base e delle classi ActionScript di Flash Player, mentre non è possibile estendere la classe TextField o le classi statiche, quali le classi Math, Key e Mouse.

Per creare questo tipo di relazione tra due classi, è necessario utilizzare la clausola `extends` dell'istruzione `class`. Per specificare una superclasse, adottare la seguente sintassi:

```
class SubClass extends SuperClass {}
```

La classe specificata come SubClass eredita tutte le proprietà e i metodi definiti in SuperClass.

Ad esempio, è possibile creare una classe Mammifero che definisce determinate proprietà e comportamenti comuni a tutti i mammiferi. Per creare una variazione della classe Mammal, ad esempio una classe Marsupial, è necessario estendere la classe Mammal, ossia creare una sottoclasse della classe Mammal, come segue:

```
class Marsupial extends Mammal {}
```

La sottoclasse eredita tutte le proprietà e i metodi della superclasse, compresi proprietà e metodi dichiarati come privati per mezzo della parola chiave `private`.

Per ulteriori informazioni sull'estensione delle classi, consultare i seguenti argomenti:

- “[Informazioni sulla creazione di una sottoclasse](#)” a pagina 281
- “[Sostituzione di metodi e proprietà](#)” a pagina 285

Per ulteriori informazioni sui membri privati, consultare “[Informazioni su metodi e proprietà \(membri\) pubblici, privati e statici](#)” a pagina 220. Per un esempio sulla creazione di una sottoclasse, consultare “[Esempio: estensione della classe Widget](#)” a pagina 282.

## Informazioni sulla creazione di una sottoclasse

Il codice seguente definisce una classe personalizzata JukeBox che estende la classe Sound e definisce un array denominato `song_arr` e un metodo denominato `playSong()` che riproduce un brano musicale e richiama il metodo `loadSound()`, ereditato dalla classe Sound.

```
class JukeBox extends Sound {
 public var song_arr:Array = new Array("beethoven.mp3", "bach.mp3",
 "mozart.mp3");
 public function playSong(songID:Number):Void {
 super.loadSound(song_arr[songID], true);
 }
}
```

Se non si chiama `super()` nella funzione di costruzione di una sottoclasse, il compilatore genera automaticamente una chiamata alla funzione di costruzione della classe immediatamente superiore senza parametri come prima istruzione della funzione. Se la superclasse non dispone di una funzione di costruzione, il compilatore ne crea una vuota e genera una chiamata a questa funzione dalla sottoclasse. Tuttavia, se la superclasse acquisisce dei parametri nella propria definizione, è necessario creare una funzione di costruzione nella sottoclasse e chiamare la superclasse con i parametri richiesti.

In ActionScript 2.0 non è consentita l'ereditarietà multipla, ossia l'ereditarietà da più di una classe. Le classi possono però ereditare da più classi se si utilizzano istruzioni `extends` singole, come nell'esempio seguente:

```
// Non consentito
class C extends A, B {} // **Errore: Una classe non può estendere più di una classe.

// Consentito
class B extends A {}
class C extends B {}
```

Per garantire una forma limitata di ereditarietà multipla è possibile anche implementare le interfacce. Per ulteriori informazioni sulle interfacce, consultare il [Capitolo 8, “Interfacce” a pagina 293](#). Per un esempio sulla creazione di una sottoclasse, consultare [“Esempio: estensione della classe Widget” a pagina 282](#). Per ulteriori informazioni su `super`, vedere `super` statement nella *Guida di riferimento di ActionScript 2.0*.

## Esempio: estensione della classe Widget

I membri delle classi si propagano alle sottoclassi della superclasse da cui sono definiti. L'esempio successivo dimostra come creare una classe `Widget` ed estenderla (creando una sottoclasse) per creare una classe `SubWidget`.

### Per creare la classe `Widget` e la sottoclasse `SubWidget`:

1. Creare un nuovo file ActionScript e salvarlo come `Widget.as`.
2. Aggiungere al nuovo documento il codice seguente:

```
class Widget {
 public static var widgetCount:Number = 0;
 public function Widget() {
 Widget.widgetCount++;
 }
}
```

- 3.** Salvare le modifiche apportate al file ActionScript.
- 4.** Creare un nuovo file ActionScript e salvarlo come **SubWidget.as** nella stessa directory della classe Widget.

- 5.** In SubWidget.as immettere il codice seguente nella finestra Script:

```
class SubWidget extends Widget {
 public function SubWidget() {
 trace("Creazione di subwidget # "+Widget.widgetCount);
 }
}
```

- 6.** Salvare le modifiche a SubWidget.as.
- 7.** Creare un nuovo file FLA e salvarlo come **subWidgetTest.fla** nella stessa directory dei file di classe ActionScript precedenti.
- 8.** Nel file subWidgetTest.fla, immettere il codice seguente nel fotogramma 1 della linea temporale principale:

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
trace("Widget.widgetCount = " + Widget.widgetCount);
trace("SubWidget.widgetCount = " + SubWidget.widgetCount);
```

Il codice precedente crea due istanze della classe SubWidget: sw1 e sw2. Ogni chiamata alla funzione di costruzione SubWidget risale al valore corrente della proprietà statica Widget.widgetCount. Poiché la classe SubWidget è una sottoclasse della classe Widget, si può accedere alla proprietà widgetCount tramite la classe SubWidget e il compilatore riscrive il riferimento (nel codice byte e non nel file ActionScript) come Widget.widgetCount. Se si cerca di accedere alla proprietà statica widgetCount al di fuori delle istanze della classe Widget o SubWidget, come sw1 o sw2, il compilatore genera un errore.

- 9.** Salvare le modifiche apportate al documento.

**10.** Selezionare Controllo > Prova filmato per provare il documento Flash.

Il pannello Output visualizza il seguente output:

```
Creazione subwidget # 1
Creazione subwidget # 2
Widget.widgetCount = 2
SubWidget.widgetCount = 2
```

Viene prodotto questo output perché, sebbene la funzione di costruzione della classe Widget non venga mai chiamata in modo esplicito, viene in realtà chiamata dalla funzione di costruzione della classe SubWidget. Per questo motivo la funzione di costruzione della classe Widget incrementa la variabile `widgetCount` della classe Widget.

Il compilatore di ActionScript 2.0 può risolvere riferimenti a membri statici all'interno delle definizioni di classe.

Se non si specifica il nome della classe per la proprietà `Widget.widgetCount`, ma si fa riferimento solo a `widgetCount`, il compilatore di ActionScript 2.0 risolve il riferimento a `Widget.widgetCount` ed esporta correttamente la proprietà. In modo analogo, se si fa riferimento alla proprietà come `SubWidget.widgetCount`, il compilatore riscrive il riferimento (nel codice byte e non nel file ActionScript) come `Widget.widgetCount`, perché SubWidget è una sottoclasse della classe Widget.

**ATTENZIONE**

Se si cerca di accedere alla variabile statica `widgetCount` dalla classe Widget tramite le istanze `sw1` o `sw2`, Flash genera un errore secondo il quale i membri statici sono accessibili solo direttamente tramite classi.

Per migliorare la leggibilità del codice, Macromedia suggerisce di utilizzare sempre riferimenti esplicativi alle variabili membro statiche presenti nel codice, come illustrato nell'esempio seguente. Utilizzando riferimenti esplicativi è possibile identificare in modo semplice dove risiede la definizione di un membro statico.

## Sostituzione di metodi e proprietà

Quando una sottoclasse estende una superclasse, ne eredita tutti i metodi e le proprietà. Uno dei vantaggi offerti dall'uso delle classi e dalla loro estensione è la possibilità non solo di garantire nuove funzionalità a una classe esistente, ma anche di modificare le funzionalità già presenti. Nell'esempio della classe Widget creata in [“Esempio: estensione della classe Widget” a pagina 282](#), è possibile creare un nuovo metodo nella superclasse (Widget) e quindi sostituire il metodo nella sottoclasse (SubWidget) o semplicemente utilizzare il metodo ereditato dalla classe Widget. L'esempio seguente mostra come sostituire i metodi già presenti nelle classi.

### Per sostituire i metodi di una sottoclasse:

1. Creare un nuovo documento ActionScript e salvarlo come **Widget.as**.
2. In Widget.as, immettere il codice ActionScript seguente nella finestra Script:

**Nota:** se in un esempio precedente è già stata creata la classe Widget, modificarne il codice aggiungendo il metodo doSomething() come segue:

```
class Widget {
 public static var widgetCount:Number = 0;
 public function Widget() {
 Widget.widgetCount++;
 }
 public function doSomething():Void {
 trace("Widget::doSomething()");
 }
}
```

3. Salvare le modifiche apportate al documento ActionScript.

La classe Widget definisce ora una funzione di costruzione e un metodo pubblico denominato **doSomething()**.

4. Creare un nuovo file ActionScript assegnandogli il nome **SubWidget.as** e salvarlo nella stessa directory di Widget.as.



Se in [“Esempio: estensione della classe Widget” a pagina 282](#) è già stata creata la classe SubWidget, è possibile utilizzare questo file.

5. In SubWidget.as, immettere il codice ActionScript seguente nella finestra Script:

```
class SubWidget extends Widget {
 public function SubWidget() {
 trace("Creazione di subwidget # "+Widget.widgetCount);
 doSomething();
 }
}
```

- 6.** Salvare le modifiche a SubWidget.as.

Come si può notare, la funzione di costruzione della classe SubWidget chiama il metodo `doSomething()` definito nella superclasse.

- 7.** Creare un nuovo documento Flash e salvarlo come **subWidgetTest.fla** nella stessa directory dei documenti ActionScript.

- 8.** In `subWidgetTest.fla`, immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale principale:

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
```

- 9.** Salvare le modifiche apportate al documento Flash.

- 10.** Selezionare Controllo > Prova filmato per provare il documento Flash. I dati seguenti devono essere visualizzati nel pannello Output:

```
Creazione subwidget # 1
Widget::doSomething()
Creazione subwidget # 2
Widget::doSomething()
```

Questo output dimostra che la funzione di costruzione della classe SubWidget chiama la funzione di costruzione della relativa superclasse (Widget) che a sua volta incrementa la proprietà statica `widgetCount`. La funzione di costruzione di SubWidget traccia la proprietà statica della superclasse e chiama il metodo `doSomething()` che eredita dalla superclasse.

- 11.** Aprire la classe SubWidget e aggiungere un nuovo metodo denominato `doSomething()`. Modificare il codice della classe, aggiungendo il codice riportato in grassetto, affinché corrisponda a quanto segue:

```
class SubWidget extends Widget {
 public function SubWidget() {
 trace("Creazione di subwidget # "+Widget.widgetCount);
 doSomething();
 }
 public function doSomething():Void {
 trace("SubWidget::doSomething()");
 }
}
```

- 12.** Salvare le modifiche apportate al file di classe e aprire nuovamente `subwidgetTest.fla`.

- 13.** Selezionare Controllo > Prova filmato per provare il file. I dati seguenti devono essere visualizzati nel pannello Output:

```
Creazione subwidget # 1
SubWidget::doSomething()
Creazione subwidget # 2
SubWidget::doSomething()
```

L'output riportato sopra mostra che il metodo `doSomething()` della funzione di costruzione della classe `SubWidget` chiama il metodo `doSomething()` della classe corrente anziché della superclasse.

Riaprire la classe `SubWidget` e modificarne la funzione di costruzione affinché chiami il metodo della superclasse `doSomething()`, aggiungendo il codice riportato in grassetto:

```
public function SubWidget() {
 trace("Creazione di subwidget # "+Widget.widgetCount);
 super.doSomething();
}
```

Come dimostrato, è possibile aggiungere la parola chiave `super` per chiamare il metodo `doSomething()` della superclasse anziché quello della classe corrente. Per ulteriori informazioni su `super`, vedere la voce corrispondente nella Guida di riferimento di ActionScript 2.0.

- 14.** Salvare il file di classe `SubWidget` con la funzione di costruzione modificata e selezionare Controllo > Prova filmato per ripubblicare il documento Flash.

Il pannello Output visualizza il contenuto del metodo `doSomething()` della classe `Widget`.

## Uso del polimorfismo in un'applicazione

La programmazione orientata agli oggetti permette di esprimere differenze tra le singole classi con l'ausilio di una tecnica detta polimorfismo, grazie alla quale le classi possono sostituire i metodi delle relative superclassi e definirne implementazioni specializzate.

È possibile, ad esempio, creare una classe `Mammifero` che disponga dei metodi `giocare()` e `dormire()`, quindi creare sottoclassi `Gatto`, `Scimmia` e `Cane` che estendano la classe `Mammifero`. Le sottoclassi sostituiscono il metodo `giocare()` della classe `Mammifero` per riflettere le abitudini di ogni animale. `Scimmia` implementa il metodo `giocare()` per passare da un albero a un altro, `Gatto` implementa il metodo `giocare()` per far rimbalzare un gomitolo di lana, mentre `Cane` implementa il metodo `giocare()` per rincorrere un palla. Dato che la funzionalità `sleep()` è simile per tutti gli animali, si utilizza l'implementazione della superclasse. La procedura seguente dimostra questo esempio in Flash.

## Per utilizzare il polimorfismo in un'applicazione:

1. Creare un nuovo documento ActionScript e salvarlo come **Mammal.as**.

Questo documento rappresenta la classe di base per diverse classi di animali che verranno create nei passaggi seguenti.

2. In Mammal.as, immettere il codice ActionScript seguente nella finestra Script:

```
class Mammal {
 private var _gender:String;
 private var _name:String = "Mammal";

 // funzione di costruzione
 public function Mammal(gender:String) {
 this._gender = gender;
 }

 public function toString():String {
 return "[object " + speciesName + "]";
 }
 public function play():String {
 return "Caccia un altro tipo come il mio";
 }
 public function sleep():String {
 return "Chiudi gli occhi";
 }

 public function get gender():String {
 return this._gender;
 }
 public function get speciesName():String {
 return this._name;
 }
 public function set speciesName(value:String):Void {
 this._name = value;
 }
}
```

La classe precedente definisce due variabili private, `_gender` e `_name`, che vengono utilizzate per memorizzare il genere dell'animale e il tipo di mammifero. In seguito viene definita la funzione di costruzione `Mammal` che accetta un solo parametro, `gender`, utilizzato per impostare la variabile privata `_gender` definita in precedenza. Vengono inoltre specificati tre metodi pubblici aggiuntivi: `toString()`, `play()` e `sleep()`, ognuno dei quali restituisce oggetti stringa. I tre metodi finali sono getter e setter per le proprietà `_gender` e `_name` del mammifero.

**3.** Salvare il documento ActionScript.

Questa classe viene utilizzata come superclasse delle classi Cat, Dog e Monkey che verranno create tra poco. Il metodo `toString()` della classe Mammal può essere utilizzato per visualizzare una rappresentazione in formato stringa delle istanze di Mammal (o delle istanze che estendono la classe Mammal).

**4.** Creare un nuovo file ActionScript e salvarlo come **Cat.as** nella stessa directory del file di classe **Mammal.as** creato al punto 1.

**5.** In **Cat.as**, immettere il codice ActionScript seguente nella finestra Script:

```
class Cat extends Mammal {
 // funzione di costruzione
 public function Cat(gender:String) {
 super(gender);
 speciesName = "Cat";
 }

 public function play():String {
 return "Prendi un gomitolo.";
 }
}
```

In questo modo si sostituisce il metodo `play()` della superclasse Mammal. La classe Cat definisce solo due metodi, una funzione di costruzione e un metodo `play()`. In quanto estende la classe Mammal, ne eredita i metodi e le proprietà. Per ulteriori informazioni sulla sostituzione, consultare “[Sostituzione di metodi e proprietà](#)” a pagina 285.

**6.** Salvare le modifiche apportate al documento ActionScript.

**7.** Creare un nuovo documento ActionScript e salvarlo come **Dog.as** nella stessa directory dei due file di classe precedenti.

**8.** In **Dog.as**, immettere il codice ActionScript seguente nella finestra Script:

```
class Dog extends Mammal {
 // funzione di costruzione
 public function Dog(gender:String) {
 super(gender);
 speciesName = "Dog";
 }

 public function play():String {
 return "Afferra un pezzo di legno.";
 }
}
```

La struttura della classe Dog, come si può notare, è molto simile a quella della classe Cat, fatta eccezione per alcuni valori. Anche la classe Dog estende la classe Mammal e quindi ne eredita metodi e proprietà. La funzione di costruzione Dog accetta una sola proprietà, gender che viene passata alla classe principale, Mammal. Anche la variabile speciesName viene sostituita e impostata sulla stringa Dog. A sua volta, viene sostituito il metodo play() della classe principale.

9. Salvare le modifiche apportate al documento ActionScript.
10. Creare un altro documento ActionScript nella stessa directory degli altri file e salvarlo come **Monkey.as**.
11. In Monkey.as, immettere il codice ActionScript seguente nella finestra Script:

```
class Monkey extends Mammal {
 // funzione di costruzione
 public function Monkey(gender:String) {
 super(gender);
 speciesName = "Monkey";
 }

 public function play():String {
 return "Dondolati da un albero.";
 }
}
```

Analogamente alle due classi precedenti, Cat e Dog, la classe Monkey estende la classe Mammal. La funzione di costruzione della classe Monkey chiama quella della classe Mammal, passando il genere alla funzione di costruzione di Mammal e impostando speciesName sulla stringa Monkey. Anche la classe Monkey sostituisce il comportamento del metodo play().

12. Salvare le modifiche apportate al documento ActionScript.
13. Dopo aver creato le tre sottoclassi della classe Mammal, creare un nuovo documento Flash e assegnarvi il nome **mammalTest.fla**.
14. In mammalTest.fla, immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale principale:

```
var mammals_arr:Array = new Array();
this.createTextField("info_txt", 10, 10, 10, 450, 80);
info_txt.html = true;
info_txt.multiline = true;
info_txt.border = true;
info_txt.wordWrap = true;

createMammals()
createReport()
```

```

function createMammals():Void {
 mammals_arr.push(new Dog("Female"));
 mammals_arr.push(new Cat("Male"));
 mammals_arr.push(new Monkey("Female"));
 mammals_arr.push(new Mammal("Male"));
}

function createReport():Void {
 var i:Number;
 var len:Number = mammals_arr.length;
 // Visualizza le informazioni di Mammal in 4 colonne di testo HTML
 utilizzando le tabulazioni.
 info_txt.htmlText = "<textformat tabstops='[110, 200, 300]'>";
 info_txt.htmlText += "Mammal\tGender\tSleep\tPlay";
 for (i = 0; i < len; i++) {
 info_txt.htmlText += "<p>" + mammals_arr[i].speciesName
 + "\t" + mammals_arr[i].gender
 + "\t" + mammals_arr[i].sleep()
 + "\t" + mammals_arr[i].play() + "</p>";
 // L'istruzione trace chiama il metodo Mammal.toString().
 trace(mammals_arr[i]);
 }
 info_txt.htmlText += "</textformat>";
}

```

Il codice di `mammalTest.fla` è leggermente più complesso rispetto a quello delle classi precedenti. In primo luogo vengono importate le tre classi degli animali.

15. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento.

Le informazioni di `Mammal` vengono visualizzate in un campo di testo sullo stage e il seguente testo viene visualizzato nel pannello Output:

```
[object Dog]
[object Cat]
[object Monkey]
[object Mammal]
```



## CAPITOLO 8

# Interfacce

Nella programmazione orientata agli oggetti (OOP, Object-Oriented Programming) un'interfaccia è un documento che permette di dichiarare (ma non definire) i metodi che devono essere inclusi in una classe. Se si lavora in team con altri sviluppatori o si creano applicazioni di grandi dimensioni in Flash, le interfacce possono essere molto utili perché consentono agli sviluppatori di identificare facilmente i metodi di base delle classi ActionScript. Questi metodi devono essere implementati quando gli sviluppatori utilizzano ogni interfaccia.

Questo capitolo illustra alcune interfacce di esempio. Al termine del capitolo l'utente sarà in grado di creare file di interfacce personalizzate. Se non si ha familiarità con la creazione di classi, consultare il [Capitolo 6, “Classi”](#) prima di iniziare le esercitazioni e gli esempi di questo capitolo.

Per ulteriori informazioni sull'uso delle interfacce, consultare le seguenti sezioni:

<a href="#">Informazioni sulle interfacce</a> .....	.293
<a href="#">Creazione di interfacce come tipi di dati</a> .....	299
<a href="#">Nozioni fondamentali sull'ereditarietà e le interfacce</a> .....	301
<a href="#">Esempio: uso di interfacce</a> .....	303
<a href="#">Esempio: creazione di un'interfaccia complessa</a> .....	305

## Informazioni sulle interfacce

Nella programmazione orientata agli oggetti, le interfacce sono come classi i cui metodi non vengono implementati (definiti) e pertanto non possono eseguire azioni. Un'interfaccia è costituita pertanto da metodi "vuoti". Un'altra classe può implementare i metodi dichiarati dall'interfaccia. In ActionScript, la distinzione tra interfaccia e oggetto vale solo per la verifica degli errori in fase di compilazione e l'applicazione delle regole del linguaggio.

Un'interfaccia non è una classe, tuttavia questo non corrisponde sempre a verità in ActionScript in fase di runtime perché l'interfaccia è astratta. Le interfacce di ActionScript esistono in fase di runtime per consentire l'inserimento dei tipi (modifica da un tipo di dati esistenti a un tipo diverso). Il modello a oggetti di ActionScript 2.0 non supporta l'ereditarietà multipla, pertanto una classe può ereditare da una sola classe principale che può essere una classe Flash di base o di Flash Player oppure definita dall'utente (personalizzata). Le interfacce possono essere utilizzate per implementare una forma limitata di ereditarietà multipla e consentire a una classe di ereditare da più classi.

In C++, ad esempio, oltre alla classe Mammifero, la classe Gatto potrebbe estendere la classe Felice, che dispone dei metodi `muovereCoda()` e `mangiareBocconcini()`. Analogamente a quanto avviene in Java, in ActionScript 2.0 una classe non può estendere più classi direttamente, ma può estendere solo una classe e implementare più interfacce. È possibile quindi creare un'interfaccia Playful che dichiari i metodi `chaseTail()` e `eatCatNip()`. Una classe Gatto, oppure una qualsiasi altra classe, può implementare questa interfaccia e fornire le definizioni per tali metodi.

Un'interfaccia può anche essere considerata un "contratto di programmazione", utilizzabile per creare relazioni tra classi altrimenti non correlate. Ad esempio, si consideri una situazione di lavoro con un team di programmatore, ciascuno dei quali si occupa di una classe diversa della stessa applicazione. In fase di progettazione dell'applicazione, viene stabilito un set di metodi che le classi utilizzeranno per comunicare. Viene quindi creata un'interfaccia che dichiara questi metodi, i relativi parametri e tipi restituiti. Qualsiasi classe che implementa questa interfaccia deve fornire le definizioni per tali metodi; in caso contrario si verifica un errore del compilatore. L'interfaccia è come un protocollo di comunicazione a cui devono aderire tutte le classi.

A tale scopo, è possibile creare una classe che definisca tutti questi metodi e far sì che ogni classe estenda questa superclasse o erediti dalla stessa. Tuttavia, dato che l'applicazione è costituita da classi non correlate, non è consigliabile inserire tutte le classi in una gerarchia comune. Una soluzione ottimale consiste nel creare un'interfaccia che dichiari i metodi che queste classi utilizzano per la comunicazione e consentire quindi alle suddette classi di implementare (fornire la propria definizione per) questi metodi.

Generalmente, è possibile eseguire una corretta programmazione senza l'utilizzo delle interfacce. Tuttavia, se utilizzate correttamente, le interfacce consentono di ottimizzare la realizzazione dell'applicazione rendendola più piacevole, scalabile e aggiornabile in modo semplice.

Le interfacce ActionScript esistono in fase di runtime per consentire l'inserimento dei tipi; consultare [Capitolo 10, "Informazioni sull'inserimento di oggetti" a pagina 378](#).

Un'interfaccia non è un oggetto né una classe, ma il flusso di lavoro è simile a quello utilizzato per le classi. Per ulteriori informazioni sul flusso di lavoro delle classi, consultare ["Creazione di file di classi personalizzate" a pagina 208](#). Per un'esercitazione sulla creazione di un'applicazione con le interfacce, consultare ["Esempio: uso di interfacce" a pagina 303](#).

Per ulteriori informazioni sull'uso delle interfacce, consultare le seguenti sezioni:

- ["Informazioni sulla parola chiave interface" a pagina 295](#)
- ["Informazioni sull'assegnazione di nomi alle interfacce" a pagina 296](#)
- ["Definizione e implementazione delle interfacce" a pagina 296](#)

## Informazioni sulla parola chiave interface

La parola chiave `interface` definisce un'interfaccia. Un'interfaccia è simile ad una classe, con le eccezioni seguenti:

- Le interfacce contengono solo le dichiarazioni dei metodi, non le loro implementazioni; vale a dire, ogni classe che implementa un'interfaccia deve fornire un'implementazione per ciascun metodo dichiarato nell'interfaccia.
- Nella definizione di un'interfaccia sono consentiti solo membri pubblici e non membri di classe e istanza.
- Le istruzioni `get` e `set` non sono consentite nelle definizioni di interfaccia.
- Per utilizzare la parola chiave `interface`, è necessario specificare ActionScript 2.0 e Flash Player 6 o versione superiore nella scheda Flash della finestra di dialogo Impostazioni pubblicazione del file FLA.

La parola chiave `this` è supportata solo se viene utilizzata in file di script esterni e non negli script creati nel pannello Azioni.

## Informazioni sull'assegnazione di nomi alle interfacce

I nomi delle interfacce hanno la prima lettera maiuscola, come i nomi delle classi, e in genere sono rappresentati da aggettivi, ad esempio `Stampabile`. Nel seguente nome di interfaccia, `IEmployeeRecords`, viene utilizzata l'iniziale maiuscola sia per la prima parola che per le parole concatenate:

```
interface IEmployeeRecords {}
```

**NOTA**

Alcuni sviluppatori utilizzano il prefisso "I" (i maiuscola) per i nomi di interfaccia, per distinguerli da quelli delle classi. Si tratta di una buona abitudine, perché consente di riconoscere rapidamente le interfacce dalle classi.

Per ulteriori informazioni sulle convenzioni per l'assegnazione dei nomi, consultare il [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0” a pagina 791](#)

## Definizione e implementazione delle interfacce

Il processo di creazione di un'interfaccia equivale a quello di creazione di una classe. Come per le classi, è possibile definire le interfacce solo in file ActionScript esterni. Il flusso di lavoro necessario per la creazione di un'interfaccia comprende almeno le fasi seguenti:

- Definizione di un'interfaccia in un file ActionScript esterno.
- Salvataggio del file di interfaccia nella directory specificata per il percorso della classe (o nel percorso in cui Flash cerca le classi) oppure nella stessa directory del file FLA dell'applicazione
- Creazione di un'istanza della classe in un altro script, ossia un documento Flash (FLA) o un file di script esterno, oppure creazione di un'interfaccia secondaria basata sulla classe originale
- Creazione di una classe che implementi l'interfaccia in un file di script esterno

Un'interfaccia viene dichiarata con l'ausilio della parola chiave `interface`, seguita dal nome dell'interfaccia e quindi da una parentesi graffa aperta e da una chiusa (`{}`) che delimitano il corpo dell'interfaccia, come nell'esempio seguente:

```
interface IEmployeeRecords {
 // Dichiarazioni del metodo interface
}
```

Un'interfaccia può contenere solo dichiarazioni di metodi (funzioni), inclusi i parametri, i tipi di parametri e i tipi restituiti di funzioni.

Per ulteriori informazioni sulle convenzioni per la creazione di una struttura per classi ed interfacce, consultare [Capitolo 19, “Procedure ottimali e convenzioni di codifica per ActionScript 2.0”](#) a pagina 791. Per un'esercitazione sulla creazione di un'applicazione con le interfacce, consultare [“Esempio: uso di interfacce”](#) a pagina 303.

Il seguente codice dichiara ad esempio un'interfaccia denominata `IMyInterface` che contiene due metodi, `method1()` e `method2()`. Il primo metodo, `method1()`, non dispone di parametri e specifica un tipo restituito `Void`, ovvero non restituisce alcun valore. Il secondo metodo, `method2()`, dispone di un solo parametro di tipo `String` e specifica un tipo restituito `Boolean`.

#### **Per creare un'interfaccia semplice:**

1. Creare un nuovo file ActionScript e salvarlo come `IMyInterface.as`.
2. Immettere il codice ActionScript seguente nella finestra Script:

```
interface IMyInterface {
 public function method1():Void;
 public function method2(param:String):Boolean;
}
```

3. Salvare le modifiche apportate al file ActionScript.

Per utilizzare l'interfaccia all'interno di un'applicazione, è necessario innanzitutto creare una classe che implementi la nuova interfaccia.

4. Creare un nuovo file ActionScript e salvarlo come `MyClass.as` nella stessa directory in cui si trova `IMyInterface.as`.
5. Nel file di classe `MyClass`, immettere il codice ActionScript seguente nella finestra Script:

```
class MyClass {
}
```

Affinché la classe personalizzata (`MyClass`) possa utilizzare la propria interfaccia (`IMyInterface`), è necessario utilizzare la parola chiave `implements` che specifica che una classe deve definire tutti i metodi dichiarati nell'interfaccia o nelle interfacce implementate.

6. Modificare il codice ActionScript di `MyClass.as` aggiungendo quello riportato in grassetto affinché corrisponda al frammento di codice seguente:

```
class MyClass implements IMyInterface {
}
```

La parola chiave `implements` deve essere inserita dopo il nome della classe.

**7.** Fare clic sul pulsante Controlla sintassi.

Flash visualizza un errore nel pannello Output indicando che MyClass deve implementare il metodo *X* dell'interfaccia IMyInterface. Il messaggio viene visualizzato perché ogni classe che estende un'interfaccia deve definire ogni metodo elencato nel documento dell'interfaccia.

**8.** Modificare nuovamente il documento MyClass aggiungendo il codice in grassetto e scrivere codice ActionScript per i metodi method1() e method2(), come nell'esempio seguente:

```
class MyClass implements IMyInterface {
 public function method1():Void {
 // ...
 };
 public function method2(param:String):Boolean {
 // ...
 return true;
 }
}
```

**9.** Salvare il documento MyClass.as e fare clic su Controlla sintassi.

Nel pannello Output non vengono più visualizzati messaggi di errore o avvisi perché sono stati definiti i due metodi.

I file di classe non devono limitarsi a contenere i metodi pubblici definiti nel file di interfaccia. Il file di interfaccia delinea solo i metodi minimi da implementare e le relative proprietà e i tipi restituiti. Le classi che implementano un'interfaccia includono quasi sempre metodi, variabili e metodi getter e setter aggiuntivi.

I file di interfaccia non possono contenere alcuna dichiarazione o assegnazione di variabile. Le funzioni dichiarate in un'interfaccia non possono contenere parentesi graffe. La seguente interfaccia, ad esempio, non viene compilata:

```
interface IBadInterface {
 // Errore del compilatore. Dichiarazioni di variabili non consentite
 // nelle interfacce.
 public var illegalVar:String;

 // Errore del compilatore. Corpi della funzione non consentiti nelle
 // interfacce.
 public function illegalMethod():Void {
 }

 // Errore del compilatore. I metodi privati non sono consentiti nelle
 // interfacce.
 private function illegalPrivateMethod():Void;
```

```
// Errore del compilatore. I metodi getter e setter non sono consentiti
nelle interfacce.
public function get illegalGetter():String;
}
```

Per un'esercitazione che illustri le modalità di creazione di un'interfaccia complessa, consultare “[Esempio: uso di interfacce](#)” a pagina 303.

Le regole per l'assegnazione dei nomi delle interfacce e per la loro memorizzazione nei pacchetti sono uguali a quelle delle classi; consultare “[Informazioni sull'assegnazione di nomi ai file di classe](#)” a pagina 240.

## Creazione di interfacce come tipi di dati

Analogamente a una classe, un'interfaccia definisce un nuovo tipo di dati. Qualsiasi classe che implementi un'interfaccia può essere considerata come appartenente al tipo definito dall'interfaccia. Ciò risulta utile per determinare se un determinato oggetto implementa una particolare interfaccia. Si consideri, ad esempio, l'interfaccia `IMovable` creata nell'esempio seguente.

### Per creare un'interfaccia come tipo di dati:

1. Creare un nuovo documento ActionScript e salvarlo nel disco rigido come `IMovable.as`.
2. In `IMovable.as`, immettere il codice ActionScript seguente nella finestra Script:

```
interface IMovable {
 public function moveUp():Void;
 public function moveDown():Void;
}
```

3. Salvare le modifiche apportate al file ActionScript.
4. Creare un nuovo documento ActionScript e salvarlo come `Box.as` nella stessa directory in cui si trova `IMovable.as`.

In questo documento, creare una classe `Box` che implementi l'interfaccia `IMovable` creata in precedenza.

- 5.** In Box.as, immettere il codice ActionScript seguente nella finestra Script:

```
class Box implements IMovable {
 public var xPos:Number;
 public var yPos:Number;

 public function Box() {
 }

 public function moveUp():Void {
 trace("moving up");
 // Definizione del metodo
 }
 public function moveDown():Void {
 trace("moving down");
 // Definizione del metodo
 }
}
```

- 6.** Salvare le modifiche apportate al documento ActionScript.
- 7.** Creare un nuovo documento Flash, assegnarvi il nome **boxTest.fla**, e salvarlo nella stessa directory dei due documenti ActionScript precedenti.
- 8.** Selezionare il fotogramma 1 della linea temporale, aprire l'Editor di ActionScript e, nel pannello Azioni (o finestra Script), immettere il codice ActionScript seguente:
- ```
var newBox:Box = new Box();
```

Questo codice ActionScript crea un'istanza della classe Box che verrà dichiarata come variabile di tipo Box.

- 9.** Salvare le modifiche apportate al documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

In Flash Player 7 e nelle versioni successive, è possibile inserire un'espressione in un tipo di interfaccia in fase di runtime. A differenza delle interfacce Java, le interfacce ActionScript esistono in fase di runtime e consentono quindi l'inserimento dei tipi. Se l'espressione è un oggetto che implementa l'interfaccia oppure dispone di una superclasse che implementa l'interfaccia, l'oggetto viene restituito. In caso contrario, viene restituito il valore `null`. Questo comportamento è utile per verificare che un particolare oggetto implementi una determinata interfaccia. Per ulteriori informazioni su l'inserimento dei tipi, consultare Capitolo 10, “Informazioni sull'inserimento di oggetti” a pagina 378.

10. Aggiungere il codice seguente dopo il codice ActionScript di boxTest.fla:

```
if (IMovable(newBox) != null) {  
    newBox.moveUp();  
} else {  
    trace("istanza di casella di controllo non mobile");  
}
```

Questo codice ActionScript verifica se l'istanza newBox implementa l'interfaccia IMovable prima di chiamare il metodo moveUp() sull'oggetto.

11. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Dato che l'istanza Box implementa l'interfaccia IMovable, viene chiamato il metodo Box.moveUp() e nel pannello Output viene visualizzato il testo "moving up".

Per ulteriori informazioni sull'inserimento, consultare il [Capitolo 10, “Informazioni sull'inserimento di oggetti” a pagina 378](#).

Nozioni fondamentali sull'ereditarietà e le interfacce

La parola chiave extends può essere utilizzata per creare sottoclassi di un'interfaccia. Questa possibilità può rivelarsi molto utile in progetti di grandi dimensioni per cui potrebbe essere necessario estendere un'interfaccia esistente (o creare una *sottoclasse*) e aggiungere ulteriori metodi. Questi metodi devono essere definiti da tutte le classi che implementano tale interfaccia.

Quando si estendono interfacce, è necessario tenere presente che Flash visualizzerà messaggi di errore se più file di interfaccia dichiarano funzioni con gli stessi nomi, ma con parametri o tipi restituiti diversi.

L'esempio seguente dimostra come estendere un'interfaccia utilizzando la parola chiave extends.

Per estendere un'interfaccia:

1. Creare un nuovo file ActionScript e salvarlo come **Ia.as**.
2. In Ia.as, immettere il codice ActionScript seguente nella finestra Script:

```
interface Ia {  
    public function f1():Void;  
    public function f2():Void;  
}
```

3. Salvare le modifiche apportate al file ActionScript.

4. Creare un nuovo file ActionScript e salvarlo come **Ib.as** nella stessa cartella del file **Ia.as** creato al punto 1.

5. In **Ib.as**, immettere il codice ActionScript seguente nella finestra Script:

```
interface Ib extends Ia {  
    public function f8():Void;  
    public function f9():Void;  
}
```

6. Salvare le modifiche apportate al file ActionScript.

7. Creare un nuovo file ActionScript e salvarlo come **ClassA.as** nella stessa directory dei due file precedenti.

8. In **ClassA.as**, immettere il codice ActionScript seguente nella finestra Script:

```
class ClassA implements Ib {  
    // f1() e f2() sono definite in interface Ia.  
    public function f1():Void {  
    }  
    public function f2():Void {  
    }  
  
    // f8() e f9() sono definite nell'interfaccia Ib che estende Ia.  
    public function f8():Void {  
    }  
    public function f9():Void {  
    }  
}
```

9. Salvare il file di classe e fare clic sul pulsante Controlla sintassi nella parte superiore della finestra Script.

Flash non genera messaggi di errore se tutti e quattro i metodi sono definiti e corrispondono alle definizioni dei rispettivi file di interfaccia.

NOTA

Sebbene si possano implementare un numero di interfacce infinito, in ActionScript 2.0 le classi possono estendere una sola classe.

Se si desidera che **ClassA** implementi più interfacce nell'esempio precedente, separare le interfacce con una virgola. In alternativa, per una classe che estende una superclasse e implementa più interfacce si utilizzerebbe codice analogo al seguente:

```
class ClassA extends ClassB implements Ib, Ic, Id { ... }.
```

Esempio: uso di interfacce

In questa esercitazione viene creata un'interfaccia semplice che può essere riutilizzata tra molte classi diverse.

Per creare un'interfaccia:

1. Creare un nuovo file ActionScript e salvarlo come **IDocumentation.as**.
2. In **IDocumentation.as**, immettere il codice ActionScript seguente nella finestra Script:

```
interface IDocumentation {  
    public function downloadUpdates():Void;  
    public function checkForUpdates():Boolean;  
    public function searchHelp(keyword:String):Array;  
}
```

3. Salvare le modifiche apportate al file di interfaccia ActionScript.
4. Creare un nuovo file ActionScript nella stessa directory di **IDocumentation.as** e salvarlo come **FlashPaper.as**.
5. In **FlashPaper.as**, immettere il codice ActionScript seguente nella finestra Script:

```
class FlashPaper implements IDocumentation {  
}
```

6. Salvare le modifiche apportate al file ActionScript.
7. Fare clic sul pulsante Controlla sintassi per verificare la classe ActionScript.

Viene visualizzato un errore analogo al seguente messaggio:

```
**Errore** percorso\FlashPaper.as: Riga 1: la classe deve implementare  
il metodo 'checkForUpdates' dall'interfaccia 'IDocumentation'.
```

```
class FlashPaper implements IDocumentation {  
}  
Errori totali di ActionScript: 1 Errori riportati: 1
```

L'errore viene visualizzato perché la classe **FlashPaper** corrente non definisce alcuno dei metodi pubblici definiti nell'interfaccia **IDocumentation**.

- 8.** Riaprire il file di classe FlashPaper.as e modificare il codice ActionScript esistente affinché corrisponda a quanto segue:

```
class FlashPaper implements IDocumentation {  
    private static var __version:String = "1,2,3,4";  
    public function downloadUpdates():Void {  
    };  
    public function checkForUpdates():Boolean {  
        return true;  
    };  
    public function searchHelp(keyword:String):Array {  
        return []  
    };  
}
```

- 9.** Salvare le modifiche apportate al file ActionScript e fare clic nuovamente sul pulsante Controlla sintassi.

Nel pannello Output non viene più visualizzato alcun errore.

NOTA

Nel file della classe FlashPaper è possibile aggiungere il numero desiderato di variabili o metodi statici, pubblici o privati. Il file di interfaccia definisce solo un set dei metodi minimi che devono essere presenti all'interno di ogni classe che implementa l'interfaccia.

- 10.** Riaprire il documento dell'interfaccia IDocumentation e aggiungere la seguente riga di codice riportata in grassetto sotto il metodo searchHelp():

```
interface IDocumentation {  
    public function downloadUpdates():Void;  
    public function checkForUpdates():Boolean;  
    public function searchHelp(keyword:String):Array;  
    public function addComment(username:String, comment:String):Void;  
}
```

- 11.** Salvare le modifiche apportate al file di interfaccia e riaprire il documento FlashPaper.as.

- 12.** Fare clic sul pulsante Controlla sintassi. Nel pannello Output viene visualizzato un nuovo messaggio di errore:

****Errore** percorso\FlashPaper.as: Riga 1: La classe deve implementare il metodo 'addComment' dall'interfaccia 'IDocumentation'.**

```
class FlashPaper implements IDocumentation {  
    Errori totali di ActionScript: 1 Errori riportati: 1
```

Questo messaggio viene visualizzato perché il file di classe FlashPaper.as non definisce più tutte le classi delineate nel file di interfaccia. Per eliminare il messaggio di errore, aggiungere il metodo addComment() alla classe FlashPaper o rimuovere la definizione del metodo dal file di interfaccia IDocumentation.

13. Immettere il metodo seguente nella classe FlashPaper:

```
public function addComment(username:String, comment:String):Void {  
    /* Invia i parametri alla pagina sul lato server che inserisce  
    commenti nel database. */  
}
```

14. Salvare le modifiche apportate a FlashPaper.as e fare clic sul pulsante Controlla sintassi.

Non dovrebbero essere più visualizzati messaggi di errore.

Nella sezione precedente è stata creata una classe in base al file dell'interfaccia IDocumentation. In questa sezione viene creata una nuova classe che a sua volta implementa l'interfaccia IDocumentation aggiungendo anche alcuni ulteriori metodi e proprietà.

Questa esercitazione dimostra l'utilità delle interfacce: se si desidera creare un'altra classe che estenda l'interfaccia IDocumentation, è possibile identificare facilmente i metodi necessari all'interno della nuova classe.

Esempio: creazione di un'interfaccia complessa

L'esempio seguente illustra diversi modi per definire e implementare le interfacce. In questa esercitazione vengono illustrate la creazione di un file di interfaccia semplice, la scrittura di una classe che implementa più interfacce e l'estensione di interfacce da parte di altre interfacce per creare strutture di dati più complesse.

Per creare un'interfaccia complessa:

1. Creare un nuovo documento ActionScript e salvarlo come **InterfaceA.as**.
2. Creare una nuova cartella assegnandole il nome **complexInterface** e salvare InterfaceA.as in questa directory.

Tutti i file creati nel corso di questa esercitazione vengono salvati in questa directory.

3. In Interface.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: InterfaceA.as  
interface InterfaceA {  
    public function k():Number;  
    public function n(z:Number):Number;  
}
```

4. Salvare il documento ActionScript e creare un nuovo documento ActionScript assegnandogli il nome **ClassB.as**, quindi salvarlo nella directory complexInterface. ClassB.as implementa l'interfaccia InterfaceA creata in precedenza.

5. In ClassB.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: ClassB.as
class ClassB implements InterfaceA {
    public function k():Number {
        return 25;
    }
    public function n(z:Number):Number {
        return (z + 5);
    }
}
```

6. Salvare le modifiche apportate al documento ClassB.as, creare un nuovo documento Flash e salvarlo come **classbTest.fla** nella directory complexInterface.

Questo file di classe prova la classe ClassB creata in precedenza.

7. In classbTest.fla, aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Nome file: classbTest.fla
import ClassB;
var myB:ClassB = new ClassB();
trace(myB.k()); // 25
trace(myB.n(7)); // 12
```

8. Salvare le modifiche apportate al documento Flash e selezionare Controllo > Prova filmato per provare il documento Flash.

Nel pannello Output vengono visualizzati due numeri, 25 e 12, che rappresentano il risultato dei metodi **k()** e **n()** della classe ClassB.

9. Creare un nuovo file ActionScript e salvarlo come **ClassC.as** nella directory complexInterface.

Questo file di classe implementa l'interfaccia InterfaceA creata al passaggio 1.

10. In ClassC.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: ClassC.as
class ClassC implements InterfaceA {
    public function k():Number {
        return 25;
    }
    // **Errore** La classe deve implementare anche il metodo "n"
    // dell'interfaccia "InterfaceA"
}
```

Se si fa clic sul pulsante Controlla sintassi per il file di classe ClassC, Flash visualizza un messaggio di errore nel pannello Output che indica che la classe corrente deve implementare il metodo **n()** definito nell'interfaccia InterfaceA. Quando si creano classi che implementano un'interfaccia, è importante definire metodi per ogni voce dell'interfaccia.

11. Creare un nuovo documento ActionScript e salvarlo come **InterfaceB.as** nella directory complexInterface.

12. In InterfaceB.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: InterfaceB.as
interface InterfaceB {
    public function o():Void;
}
```

13. Salvare le modifiche apportate al documento InterfaceB.as, creare un nuovo documento ActionScript e salvarlo come **ClassD.as** nella directory complexInterface.

Questa classe implementa sia l'interfaccia InterfaceA che l'interfaccia InterfaceB create nei passaggi precedenti. La classe ClassD deve includere le implementazioni di ogni metodo elencato in ognuno dei file di interfaccia.

14. In ClassD.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: ClassD.as
class ClassD implements InterfaceA, InterfaceB {
    public function k():Number {
        return 15;
    }
    public function n(z:Number):Number {
        return (z * z);
    }
    public function o():Void {
        trace("o");
    }
}
```

15. Salvare le modifiche apportate al file ClassD.as, creare un nuovo documento Flash e salvarlo come **classdTest.fla**.

Questo documento Flash prova la classe ClassD creata in precedenza.

16. In classdTest.fla, aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Nome file: classdTest.fla
import ClassD;
var myD:ClassD = new ClassD();
trace(myD.k()); // 15
trace(myD.n(7)); // 49
myD.o(); // o
```

17. Salvare le modifiche apportate al file classdTest.fla e selezionare Controllo > Prova filmato per provare il file.

Nel pannello Output dovrebbero essere visualizzati i valori 15, 49 e la lettera o, ovvero il risultato, rispettivamente, dei metodi **ClassD.k()**, **ClassD.n()** e **ClassD.o()**.

18. Creare un nuovo documento ActionScript e salvarlo come **InterfaceC.as**.

Questa interfaccia estende l'interfaccia InterfaceA creata in precedenza, aggiungendo la definizione di un nuovo metodo.

19. In InterfaceC.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: InterfaceC.as
interface InterfaceC extends InterfaceA {
    public function p():Void;
}
```

20. Salvare le modifiche apportate al file ActionScript, creare un nuovo file ActionScript e salvarlo come **ClassE.as** nella directory complexInterface.

Questa classe implementa due interfacce: InterfaceB e InterfaceC.

21. In ClassE.as, immettere il codice ActionScript seguente nella finestra Script:

```
// Nome file: ClassE.as
class ClassE implements InterfaceB, InterfaceC {
    public function k():Number {
        return 15;
    }
    public function n(z:Number):Number {
        return (z + 5);
    }
    public function o():Void {
        trace("o");
    }
    public function p():Void {
        trace("p");
    }
}
```

22. Salvare le modifiche apportate al documento ActionScript, creare un nuovo documento Flash e salvarlo come **classeTest.fla** nella directory complexInterface.

23. In classeTest.fla, aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Nome file: classeTest.fla
import ClassE;
var myE:ClassE = new ClassE();
trace(myE.k()); // 15
trace(myE.n(7)); // 12
myE.o(); // o
myE.p(); // p
```

24.Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Nel pannello Output vengono visualizzati i valori 15, 12, o e p, ovvero i valori restituiti dai metodi ClassE.k(), ClassE.n(), ClassE.o() e ClassE.p(). Dato che la classe ClassE implementa sia l'interfaccia InterfaceB che l'interfaccia InterfaceC, devono essere definiti tutti i metodi di entrambi i file di interfaccia. Sebbene le interfacce InterfaceB e InterfaceC definiscano solo i metodi o() e p(), InterfaceC estende InterfaceA e pertanto devono essere implementati anche tutti i metodi di questa interfaccia, ovvero k() e n().

Gestione degli eventi

Eventi: azioni che si verificano durante la riproduzione di un file SWF. Un evento quale il clic del mouse o la pressione di un tasto viene denominato *evento utente*, poiché si verifica in seguito alla diretta interazione dell'utente. Un evento generato automaticamente da Flash Player, quale l'aspetto iniziale di un clip filmato sullo stage, viene denominato *evento di sistema*, poiché non viene generato direttamente dall'utente.

Per consentire all'applicazione di reagire agli eventi, è necessario utilizzare i *gestori di eventi*, codice ActionScript associato a oggetti ed eventi specifici. Ad esempio, quando un utente seleziona un pulsante sullo stage, è possibile spostare l'indicatore di riproduzione al fotogramma successivo. In alternativa, quando viene terminato il caricamento in rete di un file XML, è possibile visualizzare il contenuto di tale file in un campo di testo.

Per informazioni su come gestire gli eventi in ActionScript, consultare le sezioni seguenti:

- “Uso dei metodi del gestore di eventi” a pagina 312
- “Uso dei listener di eventi” a pagina 314
- “Uso dei gestori di eventi pulsante e clip filmato” a pagina 319, in particolare, on handler e onClipEvent handler.
- “Trasmissione di eventi da istanze di componenti” a pagina 324

I risultati dell'uso dei gestori di eventi con loadMovie (MovieClip.loadMovie method) possono essere imprevedibili. Se si associa un gestore di eventi a un pulsante tramite `on()` oppure si crea un gestore dinamico tramite un metodo del gestore di eventi quale `onPress` (MovieClip.onPress handler), quindi si chiama `loadMovie()`, il gestore di eventi non sarà più disponibile dopo il caricamento del nuovo contenuto. Se invece si associa un gestore di eventi a un clip filmato tramite `onClipEvent` handler o `on` handler, quindi si chiama `loadMovie()` su tale clip filmato, il gestore di eventi sarà disponibile anche dopo il caricamento del nuovo contenuto.

Per ulteriori informazioni sulla gestione degli eventi, consultare le seguenti sezioni:

| | |
|---|-----|
| Uso dei metodi del gestore di eventi | 312 |
| Uso dei listener di eventi | 314 |
| Uso di listener di eventi con i componenti | 317 |
| Uso dei gestori di eventi pulsante e clip filmato | 319 |
| Trasmissione di eventi da istanze di componenti | 324 |
| Creazione di clip filmato con stati del pulsante | 324 |
| Area di validità del gestore di eventi | 325 |
| Area di validità della parola chiave this | 329 |
| Uso della classe Delegate | 329 |

Uso dei metodi del gestore di eventi

Un metodo di un gestore di eventi è un metodo di una classe che viene richiamato quando si verifica un evento su un'istanza di tale classe. La classe MovieClip, ad esempio, definisce un gestore di eventi `onPress` che viene richiamato ogni volta che si fa clic con il pulsante del mouse su un oggetto clip filmato. Contrariamente ad altri metodi di una classe, non è possibile invocare un gestore di eventi direttamente; Flash Player invoca il gestore automaticamente quando si verifica l'evento appropriato.

Le seguenti classi ActionScript sono esempi di classi che definiscono i gestori di eventi: Button, ContextMenu, ContextMenuItem, Key, LoadVars, LocalConnection, Mouse, MovieClip, MovieClipLoader, Selection, SharedObject, Sound, Stage, TextField, XML e XMLSocket. Per ulteriori informazioni sui gestori di eventi forniti dalle classi riportate in precedenza, vedere le voci della relativa classe nella *Guida di riferimento di ActionScript 2.0*. La parola *gestore* viene aggiunta nel titolo di ciascun gestore di eventi.

Per impostazione predefinita, i metodi del gestore di eventi non sono definiti: quando si verifica un evento specifico, viene chiamato il gestore di eventi corrispondente, ma l'applicazione non risponde ulteriormente all'evento. Per consentire all'applicazione di rispondere all'evento, è necessario definire una funzione con istruzione `function` e assegnare quindi tale funzione al relativo gestore di eventi. La funzione assegnata al gestore di eventi viene richiamata automaticamente ogni volta che si verifica l'evento.

Un gestore di eventi è costituito da tre elementi: l'oggetto al quale si applica l'evento, il nome del metodo del gestore di eventi dell'oggetto e la funzione assegnata al gestore di eventi. Negli esempi seguenti è illustrata la struttura di base di un gestore di eventi:

```
object.eventMethod = function () {  
    // Inserire qui il codice che risponde all'evento.  
}
```

Se, ad esempio, sullo stage è presente un pulsante denominato next_btn. Il codice seguente assegna una funzione al gestore di eventi `onPress` del pulsante, che consente di spostare l'indicatore di riproduzione al fotogramma successivo nella linea temporale:

```
next_btn.onPress = function () {
    nextFrame();
}
```

Assegnazione di un riferimento a una funzione Nel codice precedente, la funzione `nextFrame()` viene assegnata a un gestore di eventi per `onPress`. È possibile anche assegnare un riferimento (nome) di funzione a un metodo del gestore di eventi e definire la funzione successivamente, come illustrato nell'esempio seguente:

```
// Assegna un riferimento funzione al gestore di eventi onPress del
// pulsante.
next_btn.onPress = goNextFrame;

// Definisce la funzione goNextFrame().
function goNextFrame() {
    nextFrame();
}
```

Nell'esempio seguente, al gestore di eventi `onPress` viene assegnato il riferimento alla funzione e non il valore restituito dalla funzione.

```
// Errato.
next_btn.onPress = goNextFrame();
// Corretto.
next_btn.onPress = goNextFrame;
```

Ricezione di parametri passati Alcuni gestori di eventi ricevono parametri passati che forniscono informazioni relative all'evento che si è verificato. Ad esempio, il gestore di eventi `TextField.onSetFocus` viene richiamato quando l'istanza di un campo di testo viene attivata mediante tastiera. Questo gestore di eventi riceve un riferimento all'oggetto del campo di testo che è stato attivato precedentemente mediante tastiera.

Nel codice seguente, ad esempio, viene inserito un testo all'interno di un campo di testo che non è più attivato:

```
this.createTextField("my_txt", 99, 10, 10, 200, 20);
my_txt.border = true;
my_txt.type = "input";
this.createTextField("myOther_txt", 100, 10, 50, 200, 20);
myOther_txt.border = true;
myOther_txt.type = "input";
myOther_txt.onSetFocus = function(my_txt:TextField) {
    my_txt.text = "I just lost keyboard focus";
};
```

Gestori di eventi per gli oggetti di runtime È possibile inoltre assegnare funzioni ai gestori di eventi per gli oggetti creati in fase di runtime. Ad esempio, il codice seguente genera una nuova istanza di clip filmato (`newclip_mc`), quindi assegna una funzione al gestore di eventi `onPress` del filmato.

```
this.attachMovie("symbolID", "newclip_mc", 10);
newclip_mc.onPress = function () {
    trace("You pressed me");
}
```

Per ulteriori informazioni, vedere “[Creazione di clip filmato in fase di runtime](#)” a pagina 391.

Sostituzione di metodi dei gestori di eventi Se si crea una classe che estende una classe ActionScript 2.0, è possibile sostituire i metodi dei gestori di eventi con le nuove funzioni scritte. Per definire un gestore di eventi in una nuova sottoclasse e poterlo riutilizzare per diversi oggetti, collegare i simboli presenti nella libreria della classe estesa alla nuova sottoclasse. Nel codice seguente il gestore di eventi `onPress` della classe MovieClip viene sostituito con una funzione che diminuisce la trasparenza del clip filmato:

```
// Classe FadeAlpha -- imposta la trasparenza quando si fa clic sul clip
// filmato.
class FadeAlpha extends MovieClip {
    function onPress() {
        this._alpha -= 10;
    }
}
```

Per istruzioni specifiche sull'estensione di una classe ActionScript 2.0 e il suo collegamento a un simbolo della libreria, vedere gli esempi in “[Assegnazione di una classe a simboli in Flash](#)” a pagina 255. Per informazioni sulla scrittura e il lavoro con le classi personalizzate, vedere Capitolo 6, “[Classi](#)”.

Uso dei listener di eventi

I listener di eventi consentono a un oggetto, denominato *oggetto listener*, di ricevere eventi trasmessi da un altro oggetto, denominato *oggetto broadcaster*. L'oggetto broadcaster registra l'oggetto listener in modo che riceva eventi generati dall'oggetto broadcaster. È possibile, ad esempio, registrare un clip filmato in modo che riceva notifiche `onResize` dallo stage. In alternativa, un'istanza del pulsante potrebbe ricevere notifiche `onChanged` da un oggetto campo di testo. È possibile registrare più oggetti listener in modo che ricevano eventi da un singolo oggetto broadcaster ed è possibile registrare un singolo oggetto listener in modo che riceva eventi da più broadcaster.

Il modello listener/broadcaster per gli eventi, a differenza di quanto avviene con i metodi del gestore di eventi, consente di "restare in ascolto" (listen) dello stesso evento in più punti del codice senza creare conflitti. I modelli di evento diversi dal modello listener/broadcaster, ad esempio `XML.onLoad`, possono causare problemi quando lo stesso evento viene atteso in più punti del codice. Possono infatti verificarsi conflitti tra i diversi punti del codice in relazione al controllo sul riferimento all'unica funzione di callback `XML.onLoad`. Con il modello listener/broadcaster, invece, è possibile aggiungere con facilità listener allo stesso evento senza creare strozzature nel codice.

Le classi ActionScript seguenti possono trasmettere eventi: Key, Mouse, MovieClipLoader, Selection, Stage e TextField. Per verificare i listener disponibili per una classe, vedere la voce relativa a ogni classe nella *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni sui listener di eventi, consultare i seguenti argomenti:

- ["Modello dei listener di eventi" a pagina 315](#)
- ["Esempio di listener di eventi" a pagina 316](#)

La classe Stage può trasmettere eventi. Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, stagesize.fla. Questo file di esempio dimostra il modo in cui la proprietà `Stage.scaleMode` influisce sui valori di `Stage.width` e `Stage.height` quando si ridimensiona la finestra del browser.

- In Windows, scegliere *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\StageSize*.
- In Macintosh, scegliere *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/StageSize*.

Modello dei listener di eventi

Il modello di eventi per i listener di eventi è simile a quello dei gestori di eventi (vedere ["Uso dei metodi del gestore di eventi" a pagina 312](#)), con due differenze principali:

- Il gestore di eventi viene assegnato all'oggetto listener e non all'oggetto che trasmette l'evento.
- È possibile richiamare un metodo speciale dell'oggetto broadcaster, `addListener()`, che registra l'oggetto listener in modo che questo riceva gli eventi corrispondenti.

Nel codice seguente viene rappresentato il modello del listener di eventi:

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

All'inizio del codice viene inserito l'oggetto `listenerObject` con una proprietà `eventName`. L'oggetto listener può essere rappresentato da qualsiasi oggetto, ad esempio l'istanza di un oggetto esistente, di un clip filmato o di un pulsante sullo stage, oppure da un'istanza di una classe ActionScript. Un clip filmato personalizzato potrebbe ad esempio implementare i metodi del listener dello stage. Potrebbe essere presente anche un solo oggetto listener per diversi tipi di listener.

La proprietà `eventName` è un evento che si verifica su `broadcasterObject`, il quale a sua volta trasmette l'evento a `listenerObject`. È possibile registrare più listener su un broadcaster di evento.

Occorre assegnare una funzione al listener di eventi che risponde all'evento.

Infine, chiamare il metodo `addListener()` sull'oggetto broadcaster, passando ad esso l'oggetto listener.

Per annullare la registrazione di un oggetto listener in modo che non riceva più eventi, chiamare il metodo `removeEventListener()` dell'oggetto broadcaster, passando ad esso il nome dell'evento da eliminare e l'oggetto listener.

```
broadcasterObject.removeEventListener(listenerObject);
```

Esempio di listener di eventi

Nell'esempio seguente viene illustrato come utilizzare il listener di eventi `onSetFocus` per creare un gestore di attivazione semplice per un gruppo di campi di testo di input. In questo caso, viene abilitato (visualizzato) il bordo del campo di testo attivato mediante tastiera e viene disabilitato il bordo del campo di testo che non è attivato.

Per creare un gestore di attivazione semplice mediante i listener di eventi:

1. Utilizzare lo strumento Testo per creare un campo di testo sullo stage.
2. Selezionare il campo di testo e, nella finestra di ispezione Proprietà, selezionare Input dal menu a comparsa Tipo testo, quindi selezionare l'opzione Mostra bordo intorno al testo.
3. Creare un altro campo di testo di input sotto il primo.
Assicurarsi che l'opzione Mostra bordo intorno al testo non sia selezionata per questo campo di testo. Eventualmente, creare ulteriori campi di testo di input.
4. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
5. Per creare un oggetto che ascolta le notifiche di attivazione dalla classe Selection, inserire il codice seguente nel pannello Azioni:

```
// Crea l'oggetto listener focusListener.  
var focusListener:Object = new Object();
```

```

// Definisce la funzione dell'oggetto listener.
focusListener.onSetFocus = function(oldFocus_txt:TextField,
    newFocus_txt:TextField) {
    oldFocus_txt.border = false;
    newFocus_txt.border = true;
}

```

Questo codice crea un oggetto denominato `focusListener` che definisce una proprietà `onSetFocus` alla quale assegna una funzione. La funzione accetta due parametri: un riferimento al campo di testo disattivato e un riferimento al campo di testo attivato. Imposta la proprietà `border` del campo di testo disattivato su `false` e la proprietà `border` del campo di testo attivato su `true`.

- Per registrare l'oggetto `focusListener` in modo che riceva eventi dall'oggetto `Selection`, aggiungere il codice seguente nel pannello Azioni:

```

// Registra focusListener con broadcaster.
Selection.addListener(focusListener);

```

- Eseguire una prova dell'applicazione (Controllo > Prova filmato), fare clic sul primo campo di testo e premere il tasto Tab per attivare o disattivare i campi.

Uso di listener di eventi con i componenti

Quando si lavora con i componenti, la sintassi del listener di eventi è leggermente diversa. I componenti generano gli eventi, i quali devono essere specificatamente intercettati mediante un oggetto `listener` o una funzione personalizzata.

L'esempio seguente mostra come utilizzare i listener di eventi per monitorare l'avanzamento dello scaricamento di un'immagine caricata in modo dinamico.

Per intercettare gli eventi del componente Loader:

- Trascinare un'istanza del componente `Loader` dal pannello Componenti nello stage.
- Nella finestra di ispezione Proprietà, selezionare il loader e digitare `my_ldr` nella casella di testo Nome istanza.
- Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```

System.security.allowDomain("http://www.helpexamples.com");

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object):Void {
    trace(evt_obj.type); // progress
    trace("\t" + evt_obj.target.bytesLoaded + " of " +
        evt_obj.target.bytesTotal + " bytes loaded");
}
loaderListener.complete = function(evt_obj:Object):Void {
    trace(evt_obj.type); // complete
}

```

```
}

my_ldr.addEventListener("progress", loaderListener);
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");

Questo codice ActionScript definisce un oggetto listener denominato loaderListener che intercetta due eventi: progress e complete. Quando questi eventi vengono inviati, il relativo codice viene eseguito e nel pannello Output viene visualizzato il testo per il debugging (se il file SWF viene provato nello strumento di creazione).
```

Quindi, si indica all'istanza `my_ldr` di intercettare ognuno dei due eventi specificati (`progress` e `complete`) e si specifica l'oggetto listener o la funzione da eseguire quando l'evento viene inviato. Infine, viene chiamato il metodo `Loader.load()`, che attiva l'immagine da cominciare a scaricare.

4. Selezionare Controllo > Prova filmato per provare il file SWF.

L'immagine viene scaricata nell'istanza `Loader` sullo stage e diversi messaggi vengono visualizzati nel pannello Output. A seconda delle dimensioni dell'immagine che viene scaricata, e a seconda che l'immagine sia stata archiviata nella memoria cache del sistema locale dell'utente, l'evento `progress` potrebbe essere inviato diverse volte, mentre l'evento `complete` viene inviato solo quando l'immagine è stata completamente scaricata.

Quando si lavora con i componenti e si inviano gli eventi, la sintassi è leggermente diversa rispetto ai listener di eventi degli esempi precedenti. È importante ricordare, soprattutto, che è necessario utilizzare il metodo `addEventListener()` invece di chiamare `addListener()`. Inoltre, è necessario indicare l'evento specifico che si desidera intercettare oltre che l'oggetto o la funzione `listener` di eventi.

Invece di utilizzare un oggetto `listener`, come nella procedura descritta in [“Uso di listener di eventi con i componenti” a pagina 317](#), è possibile sfruttare una funzione personalizzata. Il codice usato nell'esempio precedente potrebbe essere riscritto nel modo seguente:

```
System.security.allowDomain("http://www.helpexamples.com");

my_ldr.addEventListener("progress", progressListener);
my_ldr.addEventListener("complete", completeListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.png");

function progressListener(evt_obj:Object):Void {
    trace(evt_obj.type); // progress
    trace("\t" + evt_obj.target.bytesLoaded + " of " +
        evt_obj.target.bytesTotal + " bytes loaded");
}
function completeListener(evt_obj:Object):Void {
```

```
    trace(evt_obj.type); // complete  
}
```

NOTA

Negli esempi precedenti, i listener di eventi vengono sempre aggiunti prima che venga chiamato il metodo `Loader.load()`. Se si chiama il metodo `Loader.load()` prima di specificare i listener di eventi, è possibile che il caricamento venga completato prima che i listener di eventi siano stati completamente definiti. Questo significa che il contenuto potrebbe essere visualizzato e che l'evento `complete` potrebbe non essere intercettato.

Uso dei gestori di eventi pulsante e clip filmato

È possibile associare direttamente i gestori di eventi a un'istanza di pulsante o di clip filmato sullo stage utilizzando i gestori `onClipEvent()` e `on()`. Il gestore di eventi `onClipEvent()` trasmette gli eventi associati ai clip filmato, mentre il gestore `on()` gestisce gli eventi associati ai pulsanti.

Per associare un gestore di eventi a un'istanza di pulsante o di clip filmato, attivare l'istanza del pulsante o del clip filmato sullo stage facendo clic sulla stessa, quindi immettere il codice nel pannello Azioni. Il titolo del pannello Azioni riflette il codice che verrà associato al pulsante o al clip filmato: Pannello azioni - Pulsante o Pannello Azioni - Clip filmato. Per indicazioni generali sull'uso di codice associato a istanze di pulsanti o clip filmato, consultare “[Associazione di codice agli oggetti](#)” a pagina 808.

NOTA

Non confondere i gestori di eventi di pulsanti e clip filmato con gli eventi dei componenti, ad esempio `SimpleButton.click`, `UIObject.hide` e `UIObject.reveal`, che devono essere associati a istanze di componenti e sono descritti nella guida Uso dei componenti.

È possibile associare `onClipEvent()` e `on()` solo alle istanze di clip filmato posizionate sullo stage in fase di creazione. Non è possibile associare `onClipEvent()` o `on()` alle istanze di clip filmato create in runtime (ad esempio, utilizzando il metodo `attachMovie()`). Per associare gestori di eventi a oggetti creati in fase di runtime, utilizzare i metodi del gestore di eventi o i listener di eventi. (Vedere “[Uso dei metodi del gestore di eventi](#)” a pagina 312 e “[Uso dei listener di eventi](#)” a pagina 314).

NOTA

Si sconsiglia l'associazione dei gestori `onClipEvent()` e `on()`. Per contro, è necessario inserire il codice negli script di fotogramma o in un file di classe, come illustrato in questo manuale. Per ulteriori informazioni, vedere “[Uso dei metodi del gestore di eventi](#)” a pagina 312 e “[Associazione di codice agli oggetti](#)” a pagina 808.

Per ulteriori informazioni sui gestori di eventi di pulsanti e clip filmato, consultare i seguenti argomenti:

- “[Uso di on e onClipEvent con i metodi dei gestori di eventi](#)” a pagina 320
- “[Specifiche di eventi per i metodi on o onClipEvent](#)” a pagina 321
- “[Associazione o assegnazione di più gestori a un oggetto](#)” a pagina 322

Uso di on e onClipEvent con i metodi dei gestori di eventi

A volte è possibile gestire gli eventi con tecniche diverse senza che si verifichino conflitti. Se i metodi `on()` e `onClipEvent()` vengono utilizzati insieme ai metodi dei gestori di eventi definiti dall'utente, non si verifica alcuna incompatibilità.

Ad esempio, se è presente un pulsante in un file SWF, il pulsante può disporre di un gestore `on(press)` che determina la riproduzione del file e del metodo `onPress()`, per il quale l'utente definisce una funzione che determina la rotazione di un oggetto sullo stage. Quando si fa clic sul pulsante, il file SWF viene riprodotto e l'oggetto inizia a ruotare. A seconda dei tipi di eventi e al momento in cui si desidera chiamarli, per gestire gli eventi è possibile utilizzare `on()` e `onClipEvent()`, i metodi dei gestori di eventi, oppure entrambe le tecniche.

Tuttavia, l'area di validità delle variabili e degli oggetti nei gestori `on()` e `onClipEvent()` risulta diversa da quella del gestore di eventi e del listener di eventi. Vedere “[Area di validità del gestore di eventi](#)” a pagina 325.

È possibile utilizzare il gestore `on()` per creare clip filmato nei quali si verificano eventi associati ai pulsanti. Per ulteriori informazioni, vedere “[Creazione di clip filmato con stati del pulsante](#)” a pagina 324. Per informazioni sulla specifica degli eventi per `on()` e `onClipEvent()`, vedere “[Specifiche di eventi per i metodi on o onClipEvent](#)” a pagina 321.

Per utilizzare un gestore on e il gestore di eventi onPress:

1. Creare un nuovo documento Flash e salvarlo come `handlers.fla`.
2. Selezionare lo strumento Rettangolo e disegnare un grande quadrato sullo stage.
3. Selezionare lo strumento Selezione, fare doppio clic sul quadrato disegnato nello stage e premere F8 per aprire la finestra di dialogo Converti in simbolo.
4. Inserire un nome per il simbolo del quadrato, impostare il tipo su Clip filmato e fare clic su OK.
5. Assegnare al clip filmato sullo stage il nome di istanza `box_mc`.
6. Aggiungere il seguente codice ActionScript direttamente sul simbolo del clip filmato nello stage:

```
on (press) {  
    trace("on (press) {...}");  
}
```

7. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
box_mc.onPress = function() {  
    trace("box_mc.onPress = function() {...};");  
};
```

8. Selezionare Controllo > Prova filmato per provare il documento Flash.

Quando si fa clic sul simbolo del clip filmato nello stage, l'output seguente viene inviato al pannello Output:

```
on (press) {...}  
box_mc.onPress = function() {...};
```

NOTA

Si sconsiglia l'associazione dei gestori `onClipEvent()` e `on()`. Per contro, è necessario inserire il codice negli script di fotogramma o in un file di classe, come illustrato in questo manuale. Per ulteriori informazioni, vedere [“Uso dei metodi del gestore di eventi” a pagina 312](#) e [“Associazione di codice agli oggetti” a pagina 808](#).

Specifica di eventi per i metodi on o onClipEvent

Per utilizzare un gestore `on()` o `onClipEvent()`, associarlo direttamente all'istanza di un pulsante o di un clip filmato sullo stage e specificare l'evento da gestire per tale istanza. Per un elenco completo degli eventi supportati dai gestori di eventi `on()` e `onClipEvent()`, vedere `on handler` e `onClipEvent handler` nella *Guida di riferimento di ActionScript 2.0*.

Il seguente gestore di eventi `on()`, ad esempio, viene eseguito quando l'utente fa clic sul pulsante a cui esso è associato:

```
on (press) {  
    trace("Thanks for pressing me.");  
}
```

È possibile specificare due o più eventi per ogni gestore `on()`. Gli eventi devono essere separati da virgolette. Il codice ActionScript presente in un gestore viene eseguito quando si verifica uno degli eventi specificati dal gestore. Il seguente gestore `on()` associato a un pulsante, ad esempio, viene eseguito quando il puntatore del mouse viene spostato sul pulsante e quindi al di fuori di esso.

```
on (rollOver, rollOut) {  
    trace("You rolled over, or rolled out");  
}
```

È anche possibile aggiungere eventi di pressione dei tasti utilizzando i gestori `on()`. Per esempio, il codice seguente traccia una stringa quando si preme il numero 3 sulla tastiera. Selezionare un'istanza di pulsante o di clip filmato e, nel pannello Azioni, aggiungere il seguente codice:

```
on (keyPress "3") {  
    trace("You pressed 3")  
}
```

Oppure, se si desidera tracciare il momento in cui l'utente preme il tasto Invio, è possibile utilizzare il seguente formato di codice. Selezionare un'istanza di pulsante o di clip filmato e, nel pannello Azioni, aggiungere il seguente codice:

```
on (keyPress "<Enter>") {  
    trace("Enter Pressed");  
}
```

Selezionare Controllo > Prova filmato e premere il tasto Invio per visualizzare la traccia di stringa nel pannello Output. Se non appare nessuna traccia, selezionare Controllo > Disattiva scelte rapide da tastiera e provare nuovamente. Per ulteriori informazioni sull'aggiunta di interattività alla pressione dei tasti nelle applicazioni, vedere Key.

NOTA

Si sconsiglia l'associazione dei gestori `onClipEvent()` e `on()`. Per contro, è necessario inserire il codice negli script di fotogramma o in un file di classe, come illustrato in questo manuale. Per ulteriori informazioni, vedere “[Uso dei metodi del gestore di eventi](#)” a pagina 312 e “[Associazione di codice agli oggetti](#)” a pagina 808.

Associazione o assegnazione di più gestori a un oggetto

Se si desidera che vengano eseguiti script diversi quando si verificano eventi diversi, è possibile associare più gestori a un oggetto. Ad esempio, è possibile associare i seguenti gestori `onClipEvent()` alla stessa istanza del clip filmato. Il primo viene eseguito quando il clip filmato viene caricato per la prima volta (oppure viene visualizzato sullo stage); il secondo viene eseguito quando il clip filmato viene scaricato dallo stage.

```
on (press) {  
    this.unloadMovie()  
}  
onClipEvent (load) {  
    trace("I've loaded");  
}  
onClipEvent (unload) {
```

```
    trace("I've unloaded");
}
```

NOTA

Si sconsiglia l'associazione dei gestori `onClipEvent()` e `on()`. Per contro, è necessario inserire il codice negli script di fotogramma o in un file di classe, come illustrato in questo manuale. Per ulteriori informazioni, vedere “[Uso dei metodi del gestore di eventi](#)” a pagina 312 e “[Associazione di codice agli oggetti](#)” a pagina 808.

Per associare più gestori a un oggetto utilizzando il codice posizionato sulla linea temporale, vedere il prossimo esempio. Il codice associa i gestori `onPress` e `onRelease` a un'istanza di clip filmato.

Per assegnare più gestori a un oggetto:

1. Creare un nuovo documento Flash e denominarlo **assignMulti.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
this.createEmptyMovieClip("img_mc", 10);
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
    target_mc.onPress = function() {
        target_mc.startDrag();
    };
    target_mc.onRelease = function() {
        target_mc.stopDrag();
    };
}
mcListener.onLoadError = function(target_mc:MovieClip) {
    trace("error downloading image");
}
var img_mc1:MovieClipLoader = new MovieClipLoader();
img_mc1.addListener(mcListener);
img_mc1.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);
```

3. Selezionare Controllo > Prova filmato per provare il documento.

L'immagine viene caricata nell'istanza `img_mc` e i gestori di eventi `onPress()` e `onRelease()` consentono di trascinare l'immagine sullo stage.

Trasmissione di eventi da istanze di componenti

Per ogni istanza di componente è possibile specificare la modalità di gestione dell'evento. Gli eventi dei componenti vengono gestiti in modo diverso rispetto agli eventi trasmessi da oggetti ActionScript nativi.

Per ulteriori informazioni, vedere “Gestione degli eventi dei componenti” in *Uso dei componenti*.

Creazione di clip filmato con stati del pulsante

Quando si associa un gestore `on()` a un clip filmato o si assegna una funzione a uno dei gestori di eventi del mouse MovieClip per un'istanza di clip filmato, il clip filmato risponde agli eventi associati al mouse in modo analogo a un pulsante. Inoltre, è possibile creare stati automatici del pulsante (Su, Sopra e Giù) in un clip filmato aggiungendo le etichette di un fotogramma `_up`, `_over` e `_down` alla linea temporale del clip filmato.

Quando l'utente sposta il mouse sopra un clip filmato o fa clic su di esso, l'indicatore di riproduzione viene inviato sul fotogramma con l'etichetta corretta. Per stabilire l'area attiva impiegata da un clip filmato, utilizzare la proprietà `hitArea` (`MovieClip.hitArea` property).

Per creare stati del pulsante in un clip filmato:

1. Creare un nuovo documento Flash e salvarlo come **mcbbutton.fla**.
2. Utilizzare lo strumento Rettangolo e disegnare un piccolo rettangolo (circa 100 pixel di larghezza per 20 pixel di altezza) sullo stage.
3. Fare doppio clic sulla forma con lo strumento Selezione e premere F8 per aprire la finestra di dialogo Converti in simbolo.
4. Inserire il nome **mcbbutton** per il simbolo, impostare il tipo di simbolo su clip filmato e fare clic su OK.
5. Fare doppio clic sul simbolo del clip filmato nello stage per entrare nella modalità di modifica del simbolo.
6. Creare un nuovo livello nella linea temporale del clip filmato e rinominarlo **labels**.
7. Inserire un'etichetta del fotogramma `_up` nella finestra di ispezione Proprietà.
8. Creare un nuovo livello sopra quello predefinito e un livello **labels**.

9. Rinominare il nuovo livello **actions** e aggiungere al fotogramma 1 della linea temporale del clip filmato il seguente codice ActionScript:

```
stop();
```
10. Selezionare il fotogramma 10, tutti i tre livelli, quindi selezionare Inserisci > Linea temporale > Fotogramma chiave.
11. Aggiungere un'azione `stop()` al fotogramma 10 del livello actions e inserire un'etichetta di fotogramma `_over` nel fotogramma 10 del livello labels.
12. Selezionare il rettangolo sul fotogramma 10 e utilizzare la finestra di ispezione Proprietà per selezionare un colore di riempimento diverso.
13. Creare un nuovo fotogramma chiave sul fotogramma 20 di ciascuno dei tre livelli e aggiungere un'etichetta di fotogramma `_down` nella finestra di ispezione Proprietà.
14. Modificare il colore del rettangolo nel fotogramma 20 così che ciascuno dei tre stati del pulsante abbia un colore diverso.
15. Ritornare sulla linea temporale principale.
16. Per consentire al clip filmato di rispondere agli eventi associati al mouse, eseguire una delle seguenti operazioni:

 - Associare un gestore di eventi `on()` all'istanza del clip filmato, come descritto in “[Uso dei gestori di eventi pulsante e clip filmato](#)” a pagina 319.
 - Assegnare una funzione a uno dei gestori di eventi del mouse (`onPress`, `onRelease` e così via) dell'oggetto clip filmato, come descritto in “[Uso dei metodi del gestore di eventi](#)” a pagina 312.
17. Selezionare Controllo > Prova filmato per provare il documento Flash.

Spostare il puntatore del mouse sull'istanza del clip filmato nello stage e il clip filmato si trasforma automaticamente nello stato `_over`. Fare clic sull'istanza del clip filmato e l'indicatore di riproduzione si trasforma automaticamente nello stato `_down` del clip filmato.

Area di validità del gestore di eventi

L'area di validità o *contesto* delle variabili e dei comandi dichiarati ed eseguiti all'interno di un gestore di eventi dipende dal tipo di gestore di eventi utilizzato: gestori di eventi o listener di eventi oppure gestori di eventi `on()` e `onClipEvent()`. Se si definisce un gestore di eventi in una nuova classe ActionScript, l'area di validità dipende anche dalla modalità di definizione del gestore di eventi. In questa sezione sono contenuti esempi di codice ActionScript 1.0 e ActionScript 2.0.

Esempi di ActionScript 1.0 Le funzioni assegnate ai metodi dei gestori di eventi e ai listener di eventi e le funzioni ActionScript scritte dall'utente, a differenza dei gestori `on()` e `onClipEvent()`, definiscono un'area di validità locale per le variabili.

Ad esempio, prendere in considerazione i due seguenti gestori di eventi. Il primo è un gestore di eventi `onPress` associato al clip filmato denominato `clip_mc`. Il secondo è un gestore `on()` associato alla stessa istanza del clip filmato.

```
// Associato alla linea temporale del clip principale di clip_mc:  
clip_mc.onPress = function () {  
    var shoeColor; // Variabile della funzione locale  
    shoeColor = "blue";  
}  
// Gestore on() associato a clip_mc:  
on (press) {  
    var shoeColor; // Nessuna area di validità locale per la variabile  
    shoeColor = "blue";  
}
```

Sebbene comprendano lo stesso codice, i due gestori di eventi generano risultati diversi. Nel primo caso, la variabile `color` risulta locale rispetto alla funzione definita per `onPress`. Nel secondo caso, poiché il gestore `on()` non definisce un'area di validità locale per la variabile, questa viene definita nell'area di validità della linea temporale del clip filmato `clip_mc`.

Per i gestori di eventi `on()` associati ai pulsanti, anziché ai clip filmato, le variabili (nonché le chiamate di metodi e funzioni) vengono chiamate nell'area di validità della linea temporale che contiene l'istanza del pulsante.

Ad esempio, il gestore di eventi `on()` seguente genera risultati diversi a seconda che sia associato a un oggetto clip filmato o a un oggetto pulsante. Nel primo caso, la chiamata della funzione `play()` avvia l'indicatore di riproduzione della linea temporale che contiene il pulsante. Nel secondo caso, la chiamata della funzione `play()` avvia la linea temporale del clip filmato al quale è associato il gestore.

```
// Associato al pulsante.  
on (press) {  
    play(); // Riproduce la linea temporale principale.  
}  
// Associato al clip filmato.  
on (press) {  
    play(); // Riproduce la linea temporale del clip filmato.  
}
```

Quando è associata a un oggetto pulsante, la funzione `play()` si applica alla linea temporale che contiene il pulsante, ovvero la linea temporale principale del pulsante. Quando invece il gestore `on(press)` è associato a un oggetto clip filmato, la chiamata della funzione `play()` si applica al clip filmato associato al gestore. Se il codice seguente viene associato a un clip filmato, viene riprodotta la linea temporale principale:

```
// Associato al clip filmato.
on (press) {
    _parent.play(); // Riproduce la linea temporale principale.
}
```

All'interno di un gestore di eventi o di una definizione di listener di eventi, la stessa funzione `play()` si applica alla linea temporale che contiene la definizione della funzione. Nell'esempio seguente, il metodo del gestore di eventi `my_mc.onPress` è stato dichiarato nella linea temporale che contiene l'istanza del clip filmato `my_mc`:

```
// Funzione definita in una linea temporale
my_mc.onPress = function () {
    play(); // riproduce la linea temporale su cui è definita.
};
```

Se si riproduce il clip filmato che definisce il gestore di eventi `onPress`, è necessario fare riferimento esplicitamente a tale clip utilizzando la parola chiave `this`, come indicato di seguito:

```
// Funzione definita nella linea temporale principale
my_mc.onPress = function () {
    this.play(); // riproduce la linea temporale del clip my_mc.
};
```

Tuttavia, lo stesso codice inserito nella linea temporale principale di un'istanza di pulsante riproduce invece la linea temporale principale:

```
my_btn.onPress = function () {
    this.play(); // riproduce la linea temporale principale
};
```

Per ulteriori informazioni sull'area di validità della parola chiave `this` nei gestori di eventi, vedere [“Area di validità della parola chiave this” a pagina 329](#).

Esempio di ActionScript 2.0 La classe `TextLoader` seguente viene utilizzata per caricare un file di testo e visualizza del testo dopo il caricamento del file.

```
// TextLoader.as
class TextLoader {
    private var params_lv:LoadVars;
    public function TextLoader() {
        params_lv = new LoadVars();
        params_lv.onLoad = onLoadVarsDone;
        params_lv.load("http://www.helpexamples.com/flash/params.txt");
    }
    private function onLoadVarsDone(success:Boolean):Void {
        _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
        _level0.my_txt.autoSize = "left";
        _level0.my_txt.text = params_lv.monthNames; // undefined
    }
}
```

Questo codice non può funzionare correttamente perché è presente un problema di area di validità dei gestori di eventi. `this` si potrebbe riferire infatti sia al gestore di eventi `onLoad` che alla classe. Si potrebbe quindi supporre che il metodo `onLoadVarsDone()` venga richiamato nell'area di validità dell'oggetto `TextLoader`, mentre invece viene richiamato nell'area di validità dell'oggetto `LoadVars`, perché il metodo è stato estratto dall'oggetto `TextLoader` e associato all'oggetto `LoadVars`. L'oggetto `LoadVars` chiama quindi il gestore di eventi `this.onLoad` quando il file di testo è stato caricato, mentre la funzione `onLoadVarsDone()` viene chiamata con `this` impostato su `LoadVars`, non su `TextLoader`. L'oggetto `params_lv` si trova nell'area di validità di `this` quando viene chiamato, anche se la funzione `onLoadVarsDone()` si basa sull'oggetto `params_lv` per riferimento. Per questo motivo la funzione `onLoadVarsDone()` prevede un'istanza `params_lv.params_lv` che non esiste.

Per richiamare il metodo `onLoadVarsDone()` in modo corretto nell'area di validità dell'oggetto `TextLoader`, adottare la seguente strategia: utilizzare un valore letterale di funzione per creare una funzione anonima che chiami la funzione desiderata. L'oggetto `owner` è visibile nell'area di validità della funzione anonima e pertanto può essere utilizzato per trovare l'oggetto `TextLoader` chiamante.

```
// TextLoader.as
class TextLoader {
    private var params_lv:LoadVars;
    public function TextLoader() {
        params_lv = new LoadVars();
        var owner:TextLoader = this;
        params_lv.onLoad = function (success:Boolean):Void {
            owner.onLoadVarsDone(success);
        }
        params_lv.load("http://www.helpexamples.com/flash/params.txt");
    }
    private function onLoadVarsDone(success:Boolean):Void {
        _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
        _level0.my_txt.autoSize = "left";
        _level0.my_txt.text = params_lv.monthNames; // Gennaio, Febbraio, Marzo, ...
    }
}
```

Area di validità della parola chiave this

La parola chiave `this` si riferisce all'oggetto nell'area di validità in esecuzione. A seconda del tipo di tecnica utilizzato per il gestore di eventi, `this` può fare riferimento a oggetti diversi.

All'interno di una funzione gestore di eventi o listener di eventi, `this` fa riferimento all'oggetto che definisce il metodo del gestore di eventi o del listener di eventi. Ad esempio, nel codice seguente `this` fa riferimento a `my_mc`.

```
// Gestore di eventi onPress() associato alla linea temporale principale:  
my_mc.onPress = function () {  
    trace(this); // _level0.my_mc  
}
```

All'interno di un gestore `on()` associato a un clip filmato, `this` fa riferimento al clip filmato al quale è associato il gestore `on()`, come nel codice seguente:

```
// Associato al clip filmato denominato my_mc sulla linea temporale  
// principale  
on (press) {  
    trace(this); // _level0.my_mc  
}
```

All'interno di un gestore `on()` associato a un pulsante, `this` fa riferimento alla linea temporale che contiene il pulsante, come nel codice seguente:

```
// Associato al pulsante nella linea temporale principale  
on (press) {  
    trace(this); // _level0
```

Uso della classe Delegate

La classe `Delegate` permette di eseguire una funzione in un'area di validità specifica. Questa classe viene fornita allo scopo di poter inviare lo stesso evento a due funzioni diverse (vedere “Delega di eventi alle funzioni” in *Uso dei componenti*) e quindi di chiamare le funzioni all'interno dell'area di validità della classe che le contiene.

Quando si passa una funzione come parametro a `EventDispatcher.addEventListener()`, la funzione viene chiamata nell'area di validità dell'istanza del componente broadcaster, anziché nell'oggetto in cui viene dichiarata (vedere “Delega dell'area di validità di una funzione” in *Uso dei componenti*). Per chiamare la funzione all'interno dell'area di validità dell'oggetto in cui viene dichiarata, è possibile utilizzare `Delegate.create()`.

L'esempio seguente mostra tre metodi per "ascoltare" gli eventi per un'istanza del componente `Button`. Qualunque sia il metodo per aggiungere i listener di eventi a un'istanza del componente `Button` produce l'invio dell'evento in un'area di validità diversa.

Per utilizzare la classe Delegate per "ascoltare" gli eventi:

1. Creare un nuovo documento Flash e salvarlo come **delegate.fla**.
2. Trascinare nella libreria un componente Button dalla cartella User Interface del pannello Componenti.
In un passaggio successivo si aggiungerà e posizionerà l'istanza del pulsante sullo stage attraverso il codice ActionScript.
3. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

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

function clickHandler(eventObj:Object):Void {
    trace("[" + eventObj.type + "] event on " + eventObj.target + " instance.");
    trace("\t this -> " + this);
}

var buttonListener:Object = new Object();
buttonListener.click = function(eventObj:Object):Void {
    trace("[" + eventObj.type + "] event on " + eventObj.target + " instance.");
    trace("\t this -> " + this);
};

this.createClassObject(Button, "one_button", 10, {label:"One"});
one_button.move(10, 10);
one_button.addEventListener("click", clickHandler);

this.createClassObject(Button, "two_button", 20, {label:"Two"});
two_button.move(120, 10);
two_button.addEventListener("click", buttonListener);

this.createClassObject(Button, "three_button", 30, {label:"Three"});
three_button.move(230, 10);
three_button.addEventListener("click", Delegate.create(this,
    clickHandler));
```

Il codice precedente è stato diviso in sei sezioni (ciascuna separata da una riga vuota). La prima sezione importa la classe Button (per il componente Button) e la classe Delegate. La seconda sezione del codice definisce una funzione che deve essere chiamata quando l'utente fa clic su alcuni pulsanti. La terza sezione del codice crea un oggetto utilizzato come listener di eventi, e tale oggetto "ascolta" un unico evento, `click`.

Le tre sezioni rimanenti del codice creano ciascuna una nuova istanza del componente Button sullo stage, riposizionano l'istanza e aggiungono un listener di eventi per l'evento click. Il primo pulsante aggiunge un listener di eventi per l'evento click e passa direttamente un riferimento a una funzione del gestore click. Il secondo pulsante aggiunge un listener di eventi per l'evento click e passa un riferimento a un oggetto listener, che contiene un gestore per l'evento click. Infine, la terza funzione aggiunge un listener di eventi per l'evento click, utilizza la classe Delegate per inviare l'evento click nell'area di validità di this (dove this equivale a _level0) e passa un riferimento alla funzione del gestore click.

4. Selezionare Controllo > Prova filmato per provare il documento Flash.
5. Fare clic su ciascuna istanza del pulsante per vedere in quale area di validità viene gestito l'evento.
 - a. Fare clic sul primo pulsante nello stage per tracciare il testo seguente nel pannello Output:

```
[click] event on _level0.one_button instance.
      this -> _level0.one_button
```

Quando si fa clic sull'istanza one_button, l'area di validità di this fa riferimento all'istanza stessa del pulsante.
 - b. Fare clic sul secondo pulsante nello stage per tracciare il testo seguente nel pannello Output:

```
[click] event on _level0.two_button instance.
      this -> [object Object]
```

Quando si fa clic sull'istanza two_button, l'area di validità di this fa riferimento all'oggetto buttonListener.
 - c. Fare clic sul terzo pulsante nello stage per tracciare il testo seguente nel pannello Output:

```
[click] event on _level0.three_button instance.
      this -> _level0
```

Quando si fa clic sull'istanza three_button, l'area di validità di this fa riferimento all'area di validità specificata nella chiamata al metodo Delegate.create() o, in questo caso, a _level0.

Dati e tipi di dati

10

Questo capitolo è il primo di una serie che intende sottolineare e dimostrare alcuni concetti fondamentali di ActionScript. Presenta alcune tecniche di base per la scrittura di codice necessario a creare applicazioni complesse e illustra come manipolare i dati in un file FLA, nonché i tipi di dati con cui è possibile lavorare. Nel capitolo successivo, [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi”](#) viene illustrato l’uso della sintassi e la creazione di istruzioni ActionScript. Di seguito, [Capitolo 5, “Funzioni e metodi”](#) dimostra come utilizzare funzioni e metodi nel linguaggio ActionScript.

Per ulteriori informazioni sui dati e i tipi di dati, consultare le sezioni seguenti:

| | |
|---|-----|
| Informazioni sui dati | 333 |
| Informazioni sui tipi di dati | 334 |
| Informazioni sulle variabili | 350 |
| Organizzazione di dati in oggetti | 375 |
| Informazioni sull’inserimento | 378 |

Informazioni sui dati

Per *dati* si intendono numeri, stringhe e altre informazioni modificabili all'interno di Flash. L'uso dei dati è essenziale per creare applicazioni o siti Web. I dati sono inoltre necessari per la creazione di grafica avanzata e animazione generata da script. Potrebbe inoltre essere necessario manipolare valori da utilizzare per gestire gli effetti.

È possibile definire dati in *variabili* all'interno di Flash oppure caricare dati da file o siti esterni tramite XML, servizi Web, classi ActionScript incorporate e così via. I dati possono essere archiviati in un database e le informazioni ad essi associate possono essere rappresentate in modi diversi in un file SWF, ad esempio le informazioni possono essere visualizzate in campi di testo o componenti e le immagini possono essere visualizzate in istanze di clip filmato.

Alcuni dei più comuni tipi di dati includono stringhe (una sequenza di caratteri, ad esempio nomi e porzioni di testo), numeri, oggetti (quali i clip filmato), valori booleani (true e false) e così via. In questo capitolo vengono inoltre presentati i tipi di dati in Flash e il loro utilizzo.

Per informazioni sui tipi di dati, vedere “[Informazioni sui tipi di dati](#)” a pagina 334. Per informazioni sulle variabili, vedere “[Informazioni sulle variabili](#)” a pagina 350.

Informazioni sui tipi di dati

Un *tipo di dati* descrive una porzione di dati e i tipi di operazioni eseguibili con gli stessi. I dati possono essere memorizzati in una variabile. I tipi di dati vengono utilizzati per la creazione di variabili, istanze di oggetti e definizioni di funzioni per assegnare il tipo di dati con cui lavorare. Per la scrittura di codice ActionScript si utilizzano tipi di dati diversi.

ActionScript 2.0 definisce diversi tipi di dati utilizzati comunemente. I tipi di dati descrivono il tipo di valore che una variabile o un elemento ActionScript può contenere. Una variabile a cui viene assegnato un tipo di dati può contenere solo un valore compreso all'interno del set di valori previsto per quel tipo di dati. Per informazioni sulle variabili, vedere “[Informazioni sulle variabili](#)” a pagina 350.

ActionScript dispone di numerosi tipi di dati di base che verranno con molta probabilità utilizzati frequentemente nelle applicazioni: Per ulteriori informazioni, vedere la tabella in “[Informazioni sui tipi di dati di base e complessi](#)” a pagina 335.

ActionScript dispone inoltre di classi principali, ad esempio Array e Date, che possono essere considerate tipi di dati complessi o di riferimento. Per ulteriori informazioni sui tipi di dati complessi e di riferimento, vedere “[Informazioni sui tipi di dati di base e complessi](#)” a pagina 335. Tutte le classi e tutti i tipi di dati sono inoltre definiti in modo completo nella *Guida di riferimento di ActionScript 2.0*.

È possibile creare classi personalizzate per le applicazioni. Tutte le classi definite tramite la dichiarazione con la parola chiave class sono considerate un tipo di dati. Per ulteriori informazioni sulle classi principali e altre classi incorporate, vedere “[Informazioni sulle classi incorporate e di primo livello](#)” a pagina 263. Per ulteriori informazioni sulla creazione di classi personalizzate, consultare il [Capitolo 6, “Classi” a pagina 197](#).

In ActionScript 2.0 quando si dichiarano le variabili, è possibile assegnare ad esse un tipo di dati, vale a dire uno dei tipi di dati principali o dati che rappresentino una classe personalizzata creata. Per ulteriori informazioni, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

Durante il debug degli script è spesso necessario determinare il tipo di dati di un'espressione o di una variabile per comprendere il motivo di un determinato comportamento. A questo scopo, utilizzare gli operatori `instanceof` e `typeof`. Vedere “[Informazioni sulla determinazione del tipo di dati](#)” a pagina 349.

È possibile eseguire una conversione da un tipo di dati a un altro in fase di runtime utilizzando una delle seguenti funzioni di conversione: `Array()`, `Boolean()`, `Number()`, `Object()` e `String()`.

È possibile trovare un file sorgente di esempio, `datatype.fla`, nella cartella Samples sul disco rigido, che illustra come utilizzare i tipi di dati in un'applicazione.

- In Windows, accedere a *Unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DataTypes*.

Informazioni sui tipi di dati di base e complessi

I valori dei tipi di dati possono essere suddivisi in due categorie principali: *di base* o *complessi*.

Un *valore di base* (o tipo di dati di base) viene memorizzato da ActionScript al livello di astrazione inferiore, pertanto le operazioni sui tipi di dati di base sono in genere più rapide ed efficienti rispetto a quelle eseguire sui tipi di dati complessi. I tipi di dati seguenti definiscono un set di uno o più valori di base: Boolean, null, Number, String e undefined.

Un *valore complesso* (o tipo di dati complesso) non è un valore primitivo ma fa riferimento a questo tipo di valori. Viene spesso chiamato tipo di dati di *riferimento*. I valori complessi appartengono al tipo di dati Object o a un tipo di dati basato su di esso. I tipi di dati che definiscono set di valori complessi includono Array, Date, Error, Function e XML. Per ulteriori informazioni su questi tipi di dati complessi, vedere le relative voci nella *Guida di riferimento di ActionScript 2.0*.

In determinate situazioni, le variabili contenenti tipi di dati di base funzionano diversamente da quelle contenenti tipi di dati di complessi. Per ulteriori informazioni, vedere “[Uso di variabili in un progetto](#)” a pagina 373.

ActionScript dispone dei seguenti tipi di dati di base utilizzabili nelle applicazioni:

| Tipi di dati | Descrizione |
|--------------|---|
| Boolean | Di base. Il tipo di dati Boolean può avere due valori: <code>true</code> e <code>false</code> . Nessun altro valore è consentito per le variabili di questo tipo. Il valore predefinito di una variabile booleana dichiarata ma non inizializzata è <code>false</code> . Per ulteriori informazioni, vedere “ Tipo di dati Boolean ” a pagina 337. |
| MovieClip | Complesso. I tipi di dati MovieClip consentono di controllare i simboli dei clip filmato tramite i metodi della classe MovieClip. Per ulteriori informazioni, vedere “ Tipo di dati MovieClip ” a pagina 339. |
| null | Di base. Il tipo di dati null contiene il valore <code>null</code> . Questo termine indica che non è presente alcun dato. È possibile assegnare il valore <code>null</code> in diverse situazioni per indicare che a una proprietà o variabile non è assegnato alcun valore. Il tipo di dati null è il tipo di dati predefinito per tutte le classi che definiscono tipi di dati complessi, ad eccezione della classe Object che ha come valore predefinito <code>undefined</code> . Per ulteriori informazioni, vedere “ Tipo di dati Null ” a pagina 340. |
| Number | Di base. Questo tipo di dati può rappresentare numeri interi, numeri interi senza segno e numeri a virgola mobile. Per memorizzare un numero a virgola mobile, includere una virgola decimale nel numero; senza la virgola il numero viene memorizzato come numero intero. Il tipo di dati Number può memorizzare valori da <code>Number.MAX_VALUE</code> (molto alto) a <code>Number.MIN_VALUE</code> (molto basso). Per ulteriori informazioni, vedere la <i>Guida di riferimento di ActionScript 2.0</i> e “ Tipo di dati Number ” a pagina 341. |
| Object | Complesso. Il tipo di dati Object è definito dalla classe Object che viene utilizzata come classe base per tutte le definizioni di classe in ActionScript e consente di organizzare gli oggetti all'interno di altri oggetti (oggetti nidificati). Per ulteriori informazioni, vedere “ Tipo di dati oggetto ” a pagina 341. |
| String | Di base. Il tipo di dati String rappresenta una sequenza di caratteri a 16 bit che può includere lettere, numeri e segni di punteggiatura. Le stringhe vengono memorizzate come caratteri Unicode, utilizzando il formato UTF-16. Un'operazione su un valore String restituisce una nuova istanza della stringa. Per ulteriori informazioni, vedere “ Tipo di dati String ” a pagina 342. |

| Tipi di dati | Descrizione |
|--------------|--|
| undefined | Di base. Il tipo di dati undefined contiene solo il valore undefined. È il valore predefinito delle istanze della classe Object. Alle variabili appartenenti alla classe Object è possibile assegnare solo il valore undefined. Per ulteriori informazioni, vedere "tipo di dati undefined" a pagina 344. |
| Void | Complesso. Il tipo di dati Void contiene solo il valore void. Questo tipo di dati può essere utilizzato per dichiarare funzioni che non restituiscono valori. Void è un tipo di dati complesso che fa riferimento al tipo di dati di base Void. Per ulteriori informazioni, vedere "Tipo di dati Void" a pagina 344. |

È possibile trovare un file sorgente di esempio, datatypes.fla, nella cartella Samples sul disco rigido, che illustra come utilizzare i tipi di dati in un'applicazione.

- In Windows, accedere a *Unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DataTypes*.

Tipo di dati Boolean

Un valore booleano può essere true o false. In ActionScript, i valori true e false vengono convertiti in 1 e 0 quando appropriato. I valori booleani vengono usati spesso in congiunzione con gli operatori logici in istruzioni di ActionScript che eseguono confronti per il controllo del flusso di uno script.

L'esempio seguente carica un file di testo in un file SWF e visualizza un messaggio nel pannello Output se il file di testo non viene caricato correttamente, oppure visualizza i parametri in caso di esito positivo del caricamento. Per ulteriori dettagli, vedere i commenti nell'esempio di codice.

```
var my_lv:LoadVars = new LoadVars();
//success è un valore booleano
my_lv.onLoad = function(success:Boolean) {
    //se success è true, visualizza monthNames
    if(success){
        trace(my_lv.monthNames);
    //se success è false, visualizza un messaggio
    } else {
        trace("Impossibile caricare il file di testo");
    }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

L'esempio seguente verifica che l'utente immetta valori in due istanze del componente TextInput. Vengono create due variabili booleane, `userNameEntered` e `isPasswordCorrect`, e se entrambe restituiscono true, alla variabile di tipo stringa `titleMessage` viene assegnato un messaggio di benvenuto.

```
// Aggiunge due componenti TextInput, un componente Label e un componente
// Button allo stage.
// Per le tre istanze dei componenti viene utilizzata la tipizzazione forte
// dei dati
var userName_t1:mx.controls.TextInput;
var password_t1:mx.controls.TextInput;
var submit_button:mx.controls.Button;
var welcome_lbl:mx.controls.Label;

// Nasconde l'etichetta
welcome_lbl.visible = false;

// Crea un oggetto listener utilizzato con il componente Button.
// Quando si fa clic sul componente Button, viene eseguita la verifica di
// nome utente e password
var btnListener:Object = new Object();
btnListener.click = function(evt:Object) {
    // Verifica che l'utente immetta almeno un carattere nelle istanze di
    // TextInput e restituisce un valore true o false di tipo booleano.
    var userNameEntered:Boolean = (userName_t1.text.length > 0);
    var isPasswordCorrect:Boolean = (password_t1.text == "vertigo");
    if (userNameEntered && isPasswordCorrect) {
        var titleMessage:String = "Benvenuto " + userName_t1.text + "!";
        welcome_lbl.text = titleMessage;
        // Visualizza l'etichetta
        welcome_lbl.visible = true;
    }
};
submit_button.addEventListener("click", btnListener);
```

Per ulteriori informazioni, vedere “Uso delle funzioni in Flash” a pagina 183 e “Informazioni sugli operatori logici” a pagina 162.

Tipo di dati MovieClip

I clip filmato (movie clip) sono simboli che consentono la riproduzione di animazioni in un'applicazione Flash e sono l'unico tipo di dati che si riferisce a un elemento grafico. Il tipo di dati MovieClip consente di controllare i simboli dei clip filmato tramite i metodi della classe MovieClip.

Per chiamare i metodi della classe MovieClip non vengono utilizzate le funzioni di costruzione. L'istanza di un clip filmato viene creata manualmente sullo stage o in modo dinamico. Successivamente si chiamano i metodi della classe MovieClip tramite l'operatore punto (.).

Operazioni con i clip filmato sullo stage Nell'esempio seguente vengono chiamati i metodi startDrag() e getURL() per diverse istanze di clip filmato presenti sullo stage:

```
my_mc.startDrag(true);
parent_mc.getURL("http://www.macromedia.com/support/" + product);
```

Nel secondo esempio viene restituita la larghezza di un clip filmato denominato my_mc sullo stage. L'istanza di destinazione deve essere un clip filmato e il valore restituito deve essere un valore numerico.

```
function getMCWidth(target_mc:MovieClip):Number {
    return target_mc._width;
}
trace(getMCWidth(my_mc));
```

Creazione di clip filmato in modo dinamico È comodo utilizzare ActionScript per creare clip filmato in modo dinamico se si desidera evitare di crearli sullo stage o di associarli dalla libreria. È possibile, ad esempio, creare una galleria di immagini con un numero elevato di anteprime da organizzare nello stage. Con MovieClip.createEmptyMovieClip() è possibile creare un'applicazione utilizzando esclusivamente ActionScript.

Per creare un clip filmato in modo dinamico, utilizzare

MovieClip.createEmptyMovieClip(), come indicato nel seguente esempio:

```
// Crea un clip filmato per ospitare il contenitore.
this.createEmptyMovieClip("image_mc", 9);
// Carica un'immagine in image_mc.
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Nel secondo esempio viene creato un clip filmato denominato `square_mc` in cui viene utilizzata l'API di disegno per disegnare un rettangolo. Vengono aggiunti gestori di eventi e i metodi `startDrag()` e `stopDrag()` della classe `MovieClip` per consentire di trascinare il rettangolo.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc.lineStyle(1, 0x000000, 100);
square_mc.beginFill(0xFF0000, 100);
square_mc.moveTo(100, 100);
square_mc.lineTo(200, 100);
square_mc.lineTo(200, 200);
square_mc.lineTo(100, 200);
square_mc.lineTo(100, 100);
square_mc.endFill();
square_mc.onPress = function() {
    this.startDrag();
};
square_mc.onRelease = function() {
    this.stopDrag();
};
```

Per ulteriori informazioni, vedere [Capitolo 11, “Operazioni con i clip filmato”](#) a pagina 381 e la voce `MovieClip` nella *Guida di riferimento di ActionScript 2.0*.

Tipo di dati Null

Il tipo di dati `Null` presenta un solo valore, ovvero `null`. Questo valore corrisponde a *nessun valore*, vale a dire all'assenza di dati. È possibile assegnare il valore `null` in diverse situazioni per indicare che a una proprietà o variabile non è assegnato alcun valore, ad esempio nelle situazioni seguenti:

- Per indicare che una variabile esiste ma non ha ancora ricevuto un valore
- Per indicare che una variabile esiste ma non contiene più un valore
- Come valore restituito da una funzione, per indicare che non era disponibile alcun valore che poteva essere restituito dalla funzione
- Come parametro di una funzione, per indicare che un parametro viene omesso

Diversi metodi e funzioni restituiscono `null` se non è stato impostato alcun valore.

Nell'esempio seguente è dimostrato l'utilizzo di `null` per verificare se i campi del form sono stati attivati:

```
if (Selection.getFocus() == null) {
    trace("no selection");
}
```

Tipo di dati Number

Il tipo di dati Number corrisponde a un numero a virgola mobile e precisione doppia. Il valore minimo di un oggetto numerico è di circa 5e-324, mentre il valore massimo è di circa 1.79E+308.

È possibile gestire i valori numerici tramite gli operatori aritmetici di addizione (+), sottrazione (-), moltiplicazione (*), divisione (/), modulo (%), incremento (++) e decremento (--) . Per ulteriori informazioni, vedere “[Uso degli operatori numerici](#)” a pagina 155.

Per gestire i valori numerici è inoltre possibile utilizzare i metodi delle classi incorporate Math e Number. Per ulteriori informazioni sui metodi e le proprietà di queste classi, vedere le voci Math e Number nella *Guida di riferimento di ActionScript 2.0*.

Nell'esempio seguente viene usato il metodo `sqrt()` (radice quadrata) della classe Math per restituire la radice quadrata del numero 100:

```
Math.sqrt(100);
```

Nell'esempio seguente viene tracciato un numero intero a caso compreso tra 10 e 17 (incluso):

```
var bottles:Number = 0;  
bottles = 10 + Math.floor(Math.random() * 7);  
trace("There are " + bottles + " bottles");
```

Nell'esempio seguente viene rilevata la percentuale di caricamento del clip filmato `intro_mc` che viene rappresentata sotto forma di numero intero:

```
var percentLoaded:Number = Math.round((intro_mc.getBytesLoaded() /  
    intro_mc.getBytesTotal()) * 100);
```

Tipo di dati oggetto

Un oggetto è un insieme di proprietà. Una *proprietà* è un attributo che descrive l'oggetto. La trasparenza di un oggetto, quale un clip filmato, ad esempio, è un attributo che descrive l'aspetto dell'oggetto. `_alpha` (trasparenza) è quindi una proprietà. Ogni proprietà è provvista di un nome e di un valore. Il valore di una proprietà può essere qualsiasi tipo di dati di Flash, anche il tipo di dati Object. Di conseguenza è possibile disporre oggetti all'interno di altri oggetti, ovvero *nidificarli*.

Per specificare gli oggetti e le relative proprietà, usare l'operatore punto (.). Nel codice seguente, ad esempio, `hoursWorked` è una proprietà di `weeklyStats`, che è a sua volta una proprietà di `employee`:

```
employee.weeklyStats.hoursWorked
```

L'oggetto MovieClip di ActionScript dispone di metodi che consentono di controllare le istanze del simbolo `clip` filmato nello stage. In questo esempio sono utilizzati i metodi `play()` e `nextFrame()`:

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

È inoltre possibile creare oggetti personalizzati per organizzare le informazioni nell'applicazione Flash. Per aggiungere contenuto interattivo a un'applicazione con ActionScript, occorrono diverse informazioni: ad esempio, potrebbero essere necessari il nome, l'età e il numero di telefono dell'utente, la velocità di un pallone, il nome degli articoli contenuti in un carrello, il numero di fotogrammi caricati o l'ultimo tasto premuto. La creazione di oggetti personalizzati consente di organizzare le informazioni in gruppi e di semplificare e riutilizzare gli script.

Nel codice ActionScript seguente è contenuto un esempio dell'uso di oggetti personalizzati per l'organizzazione di informazioni. Vengono creati un nuovo oggetto denominato `user` e tre proprietà: `name`, `age` e `phone` che sono tipi di dati String e Numeric.

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

Per ulteriori informazioni, vedere [“Esempio: creazione di classi personalizzate”](#) a pagina 237.

Tipo di dati String

Una stringa è una sequenza di caratteri quali lettere, numeri e caratteri di punteggiatura. Per immettere stringhe in un'istruzione ActionScript, racchiuderle tra virgolette singole ('') o doppie ("").

Il tipo di dati stringa viene spesso utilizzato per assegnare una stringa a una variabile.

Nell'istruzione seguente, ad esempio, "L7" è una stringa assegnata alla variabile `favoriteBand_str`:

```
var favoriteBand_str:String = "L7";
```

È possibile utilizzare l'operatore di addizione (+) per *concatenare*, o unire, due stringhe. In ActionScript gli spazi all'inizio o alla fine della stringa sono considerati parte integrante della stringa. La seguente espressione comprende uno spazio dopo la virgola:

```
var greeting_str:String = "Welcome, " + firstName;
```

Per inserire virgolette nelle stringhe, farle precedere da una barra rovesciata (\). Questa operazione è detta *assegnazione di una sequenza di escape* a un carattere. In ActionScript alcuni caratteri possono essere rappresentati soltanto tramite le relative sequenze di escape. Nella tabella seguente sono elencati tutti i caratteri di escape di ActionScript:

| Sequenza di escape | Carattere |
|--------------------|---|
| \b | Backspace (ASCII 8) |
| \f | Avanzamento pagina (ASCII 12) |
| \n | Avanzamento riga (ASCII 10) |
| \r | Ritorno a capo (ASCII 13) |
| \t | Tabulazione (ASCII 9) |
| \" | Virgolette doppie |
| \' | Virgolette semplici |
| \\\ | Barra rovesciata |
| \000 - \377 | Byte specificato in formato ottale |
| \x00 - \xFF | Byte specificato in formato esadecimale |
| \u0000 - \xFFFF | Carattere Unicode a 16 bit specificato in esadecimale |

Come in Java, le stringhe in ActionScript sono immutabili. Tutte le operazioni che modificano una stringa restituiscono una nuova stringa.

La classe String è una classe incorporata di ActionScript. Per informazioni sui metodi e le proprietà della classe String, vedere la voce String nella *Guida di riferimento di ActionScript 2.0*.

tipo di dati undefined

Il tipo di dati undefined presenta solo il valore `undefined`, che viene assegnato automaticamente a una variabile a cui non è stato associato alcun valore né tramite codice né tramite interazione con l'utente.

Il valore `undefined` viene assegnato in modo automatico. A differenza di `null`, non viene assegnato dal programmatore a una variabile o a una proprietà. Viene utilizzato per verificare se una variabile è stata impostata o definita. Utilizzando questo tipo di dati è possibile scrivere codice che viene eseguito solo durante l'esecuzione dell'applicazione, come illustrato nell'esempio seguente:

```
if (init == undefined) {  
    trace("initializing app");  
    init = true;  
}
```

Se l'applicazione è composta da più fotogrammi, il codice non viene eseguito una seconda volta perché la variabile `init` non risulta più indefinita.

Tipo di dati Void

Il tipo di dati Void presenta solo il valore `void` e viene utilizzato nella definizione di una funzione per indicare che questa non restituisce un valore, come illustrato nel seguente esempio:

```
// Crea una funzione con un tipo restituito Void  
function displayFromURL(url:String):Void {}
```

Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati

Le variabili vengono utilizzate in Flash per memorizzare valori nel codice. È possibile dichiarare in modo esplicito il tipo di oggetto di una variabile nel momento in cui la variabile viene creata. Questa operazione è detta *tipizzazione forte dei dati*.

Se non si definisce in modo esplicito un elemento come tipo di dati numerico, stringa o di altro tipo, in fase di runtime Flash Player tenta di determinare il tipo di dati di un elemento durante la relativa assegnazione. Se viene assegnato un valore a una variabile, come nell'esempio seguente, Flash Player valuta l'elemento a destra dell'operatore in fase di runtime e determina che si tratta di un tipo di dati numerico:

```
var x = 3;
```

Dato che `x` non era stata dichiarata utilizzando la tipizzazione forte dei dati, il compilatore non è in grado di determinarne il tipo e pertanto la variabile `x` può avere un valore di qualsiasi tipo. Vedere “[Assegnazione di un tipo di dati](#)” a pagina 346. Un’assegnazione successiva può modificare il tipo di dati di `x`; l’istruzione `x = "hello"`, ad esempio, determina la modifica del tipo di dati di `x` in String.

In ActionScript, quando un’espressione lo richiede e se per le variabili non è stata utilizzata la tipizzazione forte dei dati, i tipi di dati di base (quali Boolean, Number, String, null, o undefined) vengono sempre convertiti automaticamente.

Questa tecnica offre diversi vantaggi in fase di compilazione. La dichiarazione dei tipi di dati (tipizzazione forte dei dati) può impedire il verificarsi di errori nel codice in fase di compilazione o può aiutare a diagnosticare eventuali errori. Per dichiarare una variabile con la tipizzazione forte dei dati, utilizzare il formato seguente:

```
var variableName:datatype;
```

NOTA

La tipizzazione forte dei dati è anche detta *tipizzazione forte* di una variabile.

Dal momento che l’attribuzione di tipi di dati non appropriati determina la generazione di errori del compilatore, la tipizzazione forte dei dati aiuta a trovare errori nel codice in fase di compilazione e consente di evitare assegnazioni di tipi di dati errati a una variabile esistente. Durante la fase di creazione, la tipizzazione forte attiva i suggerimenti sul codice nell’Editor di ActionScript. Per gli elementi visivi è necessario comunque utilizzare il nome dell’istanza come suffisso.

La tipizzazione forte dei dati consente di impedire che venga assegnato inavvertitamente a una variabile un tipo di valore non appropriato. Flash verifica la presenza di errori di mancata corrispondenza del tipo in fase di compilazione e visualizza un messaggio di errore in caso di uso di un tipo di valore scorretto. L’uso della tipizzazione forte consente pertanto di evitare l’accesso a proprietà o metodi che non fanno parte del tipo di un oggetto e di visualizzare automaticamente suggerimenti sul codice per gli oggetti nell’Editor di ActionScript.

Per ulteriori informazioni sulla creazioni di variabili, vedere “[Informazioni sulle variabili](#)” a pagina 350. Per informazioni sull’assegnazione di nomi alle variabili, vedere “[Informazioni sull’assegnazione di nomi alle variabili](#)” a pagina 356. Per ulteriori informazioni sull’assegnazione di tipi di dati e sui tipi assegnabili, vedere “[Assegnazione di un tipo di dati](#)” a pagina 346.

È possibile trovare un file sorgente di esempio, datatypes.fla, nella cartella Samples sul disco rigido, che illustra come utilizzare i tipi di dati in un'applicazione.

- In Windows, accedere a *Unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DataTypes*.

Assegnazione di un tipo di dati

È necessario assegnare tipi di dati ogni volta che si definisce una variabile, sia che si dichiari una variabile tramite la parola chiave `var`, si crei un argomento di una funzione, si imposti il tipo restituito da una funzione, o si definisca una variabile da utilizzare all'interno di un ciclo `for` o `for..in`. Per assegnare un tipo di dati, utilizzare la *sintassi dei due punti*, ovvero far seguire al nome della variabile i due punti e quindi il tipo di dati:

```
var my_mc:MovieClip;
```

Sono disponibili numerosi tipi di dati, dai tipi di dati nativi quali `Number`, `String`, `Boolean`, alle classi incorporate in Flash Player 8, ad esempio `BitmapData` e `FileReference`, alle classi personalizzate scritte dagli sviluppatori. I tipi di dati più comuni che generalmente occorre specificare sono quelli incorporati, quali `Number`, `String`, `Boolean`, `Array` o `Object`, riportati negli esempi di codice seguenti.

Per assegnare un tipo di dati specifico a un elemento, specificarne il tipo tramite la parola chiave `var` e la sintassi che segue i due punti, come nell'esempio seguente:

```
// Tipizzazione forte della variabile o dell'oggetto
var myNum:Number = 7;
var birthday:Date = new Date();

// Tipizzazione forte dei parametri
function welcome(firstName:String, age:Number) {
}

// Tipizzazione forte del parametro e del valore restituito
function square(myNum:Number):Number {
    var squared:Number = myNum * myNum;
    return squared;
}
```

È possibile dichiarare il tipo di dati degli oggetti in base alle classi incorporate (`Button`, `Date` e così via) e alle classi e alle interfacce create. Se esiste, ad esempio, un file denominato `Student.as` in cui è stata definita la classe `Student`, è possibile specificare l'appartenenza degli oggetti creati al tipo `Student`:

```
var myStudent:Student = new Student();
```

Si supponga ad esempio di digitare il seguente codice:

```
// Nel file classe Student.as
class Student {
    public var status:Boolean; // Proprietà degli oggetti Student
}

// nel file FLA
var studentMaryLago:Student = new Student();
studentMaryLago.status = "enrolled"; /* Tipo non corrispondente
nell'istruzione di assegnazione: rilevato tipo String, dove era previsto
un valore Boolean */
```

Quando lo script viene compilato in Flash, viene generato il messaggio Tipo non corrispondente, perché il file SWF prevede un valore booleano.

Se si scrive una funzione che non restituisce alcun tipo, per la funzione è possibile specificare un tipo restituito Void. Se invece si crea un collegamento a una funzione, è possibile assegnare un tipo di dati Function alla nuova variabile. Per specificare che gli oggetti sono di tipo Function o Void, vedere l'esempio seguente:

```
function sayHello(name_str:String):Void {
    trace("Hello, " + name_str);
}
sayHello("world"); // Hello, world
var greeting:Function = sayHello;
greeting("Augustus"); // Hello, Augustus
```

Un ulteriore vantaggio della tipizzazione forte dei dati è rappresentato dal fatto che, se questa tecnica viene utilizzata con oggetti incorporati, Flash visualizza automaticamente suggerimenti sul codice di tali oggetti. Per ulteriori informazioni, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

I file pubblicati utilizzando ActionScript 1.0 non rispettano le assegnazioni della tipizzazione forte dei dati in fase di compilazione, pertanto il compilatore non genera un errore se viene assegnato un tipo di valore scorretto a una variabile tipizzata in modo forte.

```
var myNum:String = "abc";
myNum = 12;
/* Nessun errore in ActionScript 1.0, errore di tipo non corrispondente in
ActionScript 2.0 */
```

Questo comportamento è dovuto al fatto che, quando si pubblica un file per ActionScript 1.0, Flash interpreta un'istruzione, ad esempio `var myNum:String = "abc"`, come sintassi della barra anziché come tipizzazione forte dei dati. ActionScript 2.0 non supporta la sintassi della barra. È possibile, pertanto, che un oggetto venga assegnato a una variabile di tipo non adeguato e che, di conseguenza, il compilatore consenta, senza segnalarlo, il passaggio di chiamate di metodo non valide e di riferimenti a proprietà non definite.

La tipizzazione dei dati nei file pubblicati con ActionScript 2.0 è facoltativa. Di conseguenza, se si utilizza la tecnica della tipizzazione forte dei dati nel codice, assicurarsi che nelle impostazioni sia prevista la pubblicazione per ActionScript 2.0. Per specificare le impostazioni di pubblicazione e definire la versione di ActionScript per cui pubblicare i file, utilizzare il menu principale (File > Impostazioni pubblicazione) o fare clic sul pulsante Impostazioni nella finestra di ispezione Proprietà, facendo attenzione che non siano selezionate istanze. Per utilizzare una versione specifica di ActionScript o di Flash Player, selezionare la scheda Flash nella finestra di dialogo Impostazioni pubblicazione e selezionare l'opzione desiderata dal menu a comparsa relativo alla versione di ActionScript.

Per informazioni sulla verifica del tipo, vedere “[Informazioni sulla verifica del tipo](#)” a pagina 348.

Informazioni sulla verifica del tipo

Per *verifica del tipo* si intende il controllo della compatibilità del tipo di una variabile e di un'espressione. Flash verifica che il tipo specificato per una variabile corrisponda al valore assegnato. Per ulteriori informazioni sulla tipizzazione forte dei dati e l'assegnazione dei tipi di dati, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344 e “[Assegnazione di un tipo di dati](#)” a pagina 346.

La verifica del tipo può avvenire sia in fase di compilazione che di runtime. Se si utilizza la tipizzazione forte dei dati, la verifica del tipo viene eseguita in fase di compilazione. Dato che ActionScript è un linguaggio tipizzato in modo dinamico, ActionScript 2.0 può anche eseguire la verifica del tipo in fase di runtime.

Il codice seguente, ad esempio, non specifica il tipo di dati del parametro `xParam`. In fase di runtime, il parametro viene utilizzato per memorizzare un valore di tipo Number e quindi un valore di tipo String. La funzione `dynamicTest()` utilizza quindi l'operatore `typeof` per provare se il parametro è di tipo String o Number.

```
function dynamicTest(xParam) {
    if (typeof(xParam) == "string") {
        var myStr:String = xParam;
        trace("String: " + myStr);
    } else if (typeof(xParam) == "number") {
        var myNum:Number = xParam;
        trace("Number: " + myNum);
    }
}
dynamicTest(100);
dynamicTest("one hundred");
```

Non è necessario aggiungere informazioni sul tipo di dati in modo esplicito in ActionScript. Il compilatore ActionScript consente di utilizzare proprietà e richiamare metodi che non esistono in fase di compilazione, permettendo di creare proprietà o assegnare metodi in modo dinamico in fase di runtime.

La flessibilità garantita dalla verifica dinamica del tipo comprende l'uso di proprietà e metodi non noti in fase di compilazione. Dato che il codice è meno restrittivo, in alcune situazioni vengono assicurati diversi vantaggi. Il codice seguente, ad esempio, crea una funzione denominata `runtimeTest()` che richiama un metodo e restituisce una proprietà. Né il metodo, né la proprietà sono noti al compilatore. Il codice non genera un errore in fase di compilazione, ma se la proprietà o il metodo non è accessibile in fase di runtime, si verificherà un errore di runtime.

```
function runtimeTest(myParam) {  
    myParam.someMethod();  
    return myParam.someProperty;  
}
```

Informazioni sulla determinazione del tipo di dati

Durante le operazioni di prova e debug dei programmi, è possibile che si presentino problemi apparentemente correlati ai tipi di dati di diversi elementi. Se invece si utilizzano variabili non associate in modo esplicito a un tipo di dati, potrebbe risultare utile conoscere il tipo di dati di una determinata variabile. Con ActionScript è possibile determinare il tipo di dati di un elemento utilizzando l'operatore `typeof` per restituire informazioni sui dati.

L'operatore `typeof` consente di ottenere i tipi di dati, ma non restituisce informazioni sulla classe a cui appartiene un'istanza,

Nell'esempio seguente viene indicato come utilizzare l'operatore `typeof` per restituire il tipo di oggetto analizzato:

```
// Crea una nuova istanza della classe LoadVars  
var my_lv:LoadVars = new LoadVars();  
  
/* L'operatore typeof non specifica la classe ma indica solo che my_lv è un  
oggetto */  
var typeResult:String = typeof(my_lv);  
trace(typeResult); // Oggetto
```

In questo esempio viene creata una nuova variabile String denominata `myName` che viene convertita in un tipo di dati `Number`:

```
var myName:String = new String("17");  
trace(myName instanceof String); // true  
var myNumber:Number = new Number(myName);  
trace(myNumber instanceof Number); // true
```

Per ulteriori informazioni su questi operatori, vedere `typeof` operator e `instanceof` operator nella *Guida di riferimento di ActionScript 2.0*. Per ulteriori informazioni sulla prova e il debug, vedere [Capitolo 18, “Esecuzione del debug delle applicazioni” a pagina 771](#) Per ulteriori informazioni sull'ereditarietà e le interfacce, vedere [Capitolo 7, “Ereditarietà” a pagina 279](#). Per ulteriori informazioni sulle classi, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Informazioni sulle variabili

Una *variabile* è un contenitore di informazioni. Il codice ActionScript riportato di seguito mostra una variabile ActionScript:

```
var myVariable:Number = 10;
```

Questa variabile memorizza un valore numerico. L'uso di `:Number` nel codice riportato in precedenza permette di assegnare il tipo di valore che la variabile memorizza. Questa operazione è detta *tipizzazione dei dati*. Per ulteriori informazioni sulla tipizzazione dei dati, vedere [“Informazioni sull’assegnazione dei tipi di dati e la tipizzazione forte dei dati” a pagina 344](#) e [“Assegnazione di un tipo di dati” a pagina 346](#).

Il contenitore, rappresentato dal nome della variabile, è sempre lo stesso in tutto il codice ActionScript, ma il contenuto (il *valore*) può cambiare. Il valore di una variabile in uno script può essere modificato più volte. Quando si modifica il valore di una variabile durante la riproduzione del file SWF, è possibile registrare e salvare informazioni sulle azioni svolte dall'utente, registrare i valori che vengono modificati durante la riproduzione del file o verificare se una condizione è `true` o `false`. Potrebbe essere necessario aggiornare la variabile continuamente durante la riproduzione del file SWF, ad esempio nel caso di un gioco Flash in cui il punteggio di un giocatore cambia. Le variabili sono fondamentali quando si crea e si gestisce l'interazione con l'utente in un file SWF.

Quando si dichiara una variabile per la prima volta, è opportuno assegnarvi un valore. L'assegnazione di un valore iniziale è detta *inizializzazione* della variabile e viene spesso eseguita sul fotogramma 1 della linea temporale o dall'interno di una classe che viene caricata all'inizio della riproduzione del file SWF. Sono disponibili diversi tipi di variabili che vengono utilizzate in base all'area di validità. Per ulteriori informazioni sui diversi tipi di variabili e sulle aree di validità, vedere “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362.

SUGGERIMENTO

L'inizializzazione di una variabile consente di tenere traccia del valore della variabile e di confrontarlo durante la riproduzione del file SWF.

NOTA

Flash Player 7 e le versioni successive valutano le variabili non inizializzate in modo diverso rispetto a Flash Player 6 e alle versioni precedenti. Se si ha familiarità con gli script per Flash Player 6 e si desidera scrivere script per Flash Player 7 o una versione successiva oppure trasferire gli script esistenti a questa versione, si consiglia di analizzare attentamente queste differenze per evitare di ottenere risultati imprevisti.

Le variabili possono contenere tipi di dati diversi. Per ulteriori informazioni, vedere “[Informazioni sui tipi di dati](#)” a pagina 334. Il tipo di dati contenuto in una variabile determina come il valore della variabile viene modificato alla sua assegnazione in uno script.

I tipi di informazioni che vengono in genere memorizzati in una variabile comprendono URL (tipo String), nomi utente (tipo String), risultati di un'operazione matematica (tipo Number), numero di volte che si è verificato un evento (tipo Number) e stato di selezione di un pulsante (tipo Boolean). Ogni file SWF e ogni istanza di oggetto (ad esempio un clip filmato) sono associati a un insieme di variabili; a ciascuna variabile è assegnato un valore indipendente dalle variabili di altri file SWF o clip filmato.

Per visualizzare il valore di una variabile, utilizzare l'istruzione `trace()` per inviarne il valore al pannello Output. Il valore viene visualizzato nel pannello Output durante la prova del file SWF nell'ambiente di prova. Ad esempio, `trace(hoursWorked)` invia il valore della variabile `hoursWorked` al pannello Output nell'ambiente di prova. In tale ambiente è inoltre possibile verificare e impostare i valori delle variabili nel debugger.

Per ulteriori informazioni sulle variabili, consultare i seguenti argomenti:

- “[Informazioni sulla dichiarazione di variabili](#)” a pagina 352
- “[Informazioni sull'assegnazione di valori](#)” a pagina 353
- “[Informazioni sull'assegnazione di nomi alle variabili](#)” a pagina 356
- “[Uso di variabili in un'applicazione](#)” a pagina 357
- “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362
- “[Informazioni sui valori predefiniti](#)” a pagina 352
- “[Informazioni sugli operatori e le variabili](#)” a pagina 355
- “[Informazioni sul caricamento di variabili](#)” a pagina 367
- “[Uso di variabili in un progetto](#)” a pagina 373

Informazioni sulla dichiarazione di variabili

Le variabili possono essere dichiarate su un fotogramma nella linea temporale, direttamente su un oggetto o all'interno di un file di classe esterno.

Definire le variabili tramite la parola chiave `var` e seguire le convenzioni per l'assegnazione dei nomi alle variabili. È possibile dichiarare una variabile denominata `firstName`, come illustrato nell'esempio seguente:

```
var firstName:String;
```

Quando si dichiara una variabile, le si assegna un tipo di dati. In questo caso, viene assegnato il tipo di dati `String` alla variabile `firstName`. Per ulteriori informazioni sull'assegnazione di tipi di dati, consultare il “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

Informazioni sui valori predefiniti

Per *valore predefinito* si intende il valore che una variabile contiene prima che ne venga impostato il valore. Una variabile viene *inizializzata* quando se ne impone il valore per la prima volta. Se si dichiara una variabile ma non se ne impone il valore, tale variabile rimane *non inizializzata*. Il valore di una variabile non inizializzata corrisponde per impostazione predefinita a `undefined`. Per ulteriori informazioni sulla creazione e l'uso di variabili, vedere “[Informazioni sulle variabili](#)” a pagina 350.

Informazioni sull'assegnazione di valori

È possibile definire un *valore* come contenuto corrente di una variabile. Il valore può essere rappresentato da stringhe, numeri, array, oggetti, XML, date o anche classi personalizzate create dal programmatore. Tenere a mente che la dichiarazione di una variabile viene eseguita con la parola chiave `var`. Quando si dichiara una variabile, le si assegna anche un tipo di dati. È possibile inoltre assegnarvi un valore se questo corrisponde al tipo di dati associato alla variabile.

Nell'esempio seguente viene dimostrato come creare una variabile denominata `catName`:

```
var catName:String;
```

Dopo aver dichiarato la variabile, è possibile assegnarvi un valore. È possibile far seguire la riga precedente del codice ActionScript con la riga seguente:

```
catName = "Pirate Eye";
```



Dato che `Pirate Eye` è una stringa, il valore deve essere racchiuso tra virgolette diritte.

In questo esempio viene assegnato il valore di `Pirate Eye` alla variabile `catName`. Quando si dichiara la variabile, è possibile anche assegnarvi un valore anziché assegnarlo in seguito come negli esempi precedenti. È possibile impostare la variabile `catName` al momento della dichiarazione, come illustrato nell'esempio seguente:

```
var catName:String = "Pirate Eye";
```

Per visualizzare il valore della variabile `catName` nell'ambiente di prova, utilizzare l'istruzione `trace()`. L'istruzione invia il valore al pannello Output. È possibile analizzare il valore della variabile `catName` per vedere che il valore effettivo non comprende le virgolette tramite il seguente codice ActionScript:

```
var catName:String = "Pirate Eye";
trace(catName); // Pirate Eye
```

Tenere a mente che il valore assegnato deve corrispondere al tipo di dati assegnato, in questo caso String. Se si tenta in seguito di assegnare un numero alla variabile `catName`, ad esempio `catName = 10`, nel pannello Output viene visualizzato l'errore seguente durante la prova del file SWF:

```
Type mismatch in assignment statement: found Number where String is required.
```

Questo errore indica che si è tentato di impostare un tipo di dati scorretto per una determinata variabile.

Quando si assegna un valore numerico a una variabile, le virgolette non sono necessarie, come nel codice di esempio seguente:

```
var numWrinkles:Number = 55;
```

Se si desidera modificare il valore di numWrinkles in seguito, è possibile assegnare un nuovo valore con il seguente codice ActionScript:

```
numWrinkles = 60;
```

Quando si riassegna un valore a una variabile esistente, non occorre utilizzare la parola chiave var o definire il tipo di dati della variabile (in questo caso, :Number).

Se il valore è numerico o booleano (true o false), per il valore non devono essere utilizzate le virgolette diritte. Nel frammento di codice seguente sono presenti esempi di valori numerici e booleani:

```
var age:Number = 38;
var married:Boolean = true;
var hasChildren:Boolean = false;
```

Nell'esempio precedente, la variabile age contiene un valore intero (non decimale), ma è anche possibile utilizzare un valore in virgola mobile o decimale, ad esempio 38,4. Le variabili booleane, ad esempio married o hasChildren, possono avere solo due valori, vale a dire true o false.

Se si desidera creare un array e assegnarvi valori, il formato è leggermente diverso, come nel codice seguente:

```
var childrenArr:Array = new Array("Pylon", "Smithers", "Gil");
```

È disponibile una sintassi alternativa per la creazione di un array che prevede l'uso degli operatori di accesso agli array e delle parentesi quadre ([]). L'esempio precedente può pertanto essere riscritto come segue:

```
var childrenArr:Array = ["Pylon", "Smithers", "Gil"];
```

Per ulteriori informazioni sulla creazione di array e sugli operatori di accesso agli array, vedere “[Informazioni sugli array](#)” a pagina 129 e “[Informazioni sull'uso della sintassi del punto per identificare un'istanza](#)” a pagina 81.

Analogamente, è possibile creare un nuovo oggetto denominato myObj in uno dei modi seguenti. La prima modalità di creazione di un array, che richiede più codice, è la seguente:

```
var myObj:Object = new Object();
myObj.firstName = "Steve";
myObj.age = 50;
myObj.childrenArr = new Array("Mike", "Robbie", "Chip");
```

La seconda modalità, più breve, per la creazione dell'array myObj è la seguente:

```
var myObj:Object = {firstName:"Steve", age:50, childrenArr:["Mike",
"Robbie", "Chip"]};
```

Come illustrato nell'esempio, il metodo più breve consente di risparmiare tempo e di scrivere meno codice, in particolare per la definizione di istanze di oggetti. È importante conoscere questa sintassi alternativa perché verrà utilizzata in caso di lavoro in team o con codice ActionScript di terze parti presente, ad esempio, in Internet o in alcune pubblicazioni.

NOTA

Non tutte le variabili devono essere definite in modo esplicito. Alcune vengono create automaticamente da Flash. Per rilevare, ad esempio, le dimensioni dello stage, è possibile utilizzare i due valori predefiniti seguenti: `Stage.width` e `Stage.height`.

Informazioni sugli operatori e le variabili

I simboli matematici presenti nel codice sono detti *operatori* in ActionScript e permettono di calcolare un nuovo valore da uno o più valori e assegnare un valore a una variabile. L'operatore di uguaglianza (=) assegna un valore a una variabile:

```
var username:String = "Gus";
```

Un ulteriore esempio è l'operatore di addizione (+) che consente di sommare due o più valori numerici per produrre un nuovo valore. Utilizzando l'operatore + con due o più valori stringa, le stringhe vengono concatenate. I valori gestiti dagli operatori sono denominati *operandi*.

Quando si assegna un valore, l'operatore definisce il valore della variabile. Nello script seguente, ad esempio, viene utilizzato l'operatore di assegnazione per assegnare il valore 7 alla variabile numChildren:

```
var numChildren:Number = 7;
```

Se si desidera modificare il valore della variabile numChildren, utilizzare il codice seguente:

```
numChildren = 8;
```

NOTA

Non occorre utilizzare `var` perché la variabile è già stata definita in precedenza.

Per ulteriori informazioni sull'uso degli operatori nel codice ActionScript, vedere “[Informazioni sugli operatori](#)” a pagina 143.

Informazioni sull'assegnazione di nomi alle variabili

Prestare attenzione quando si assegnano nomi alle variabili perché, sebbene sia possibile assegnare praticamente qualsiasi nome, occorre attenersi ad alcune regole. Il nome di una variabile deve soddisfare le seguenti regole:

- Una variabile deve essere un identificatore.

NOTA

Per *identificatore* si intende il nome di una variabile, una proprietà, un oggetto, una funzione o un metodo. Il primo carattere di un identificatore deve essere costituito da una lettera, un carattere di sottolineatura (_) o dal simbolo del dollaro (\$). Ogni carattere successivo deve essere una lettera, un numero, un carattere di sottolineatura o un simbolo del dollaro.

- Una variabile non può essere una parola chiave o un valore letterale ActionScript come true, false, null o undefined. Per ulteriori informazioni sui valori letterali, vedere “[Informazioni sui valori letterali](#)” a pagina 94.
- Una variabile deve essere univoca all'interno della relativa area di validità (vedere “[Informazioni sulle variabili e l'area di validità](#)” a pagina 362).
- Una variabile non deve essere un elemento del linguaggio ActionScript, ad esempio un nome di classe.

Se per l'assegnazione dei nomi alle variabili non vengono seguite queste regole, potrebbero verificarsi errori di sintassi o risultati imprevisti. Nell'esempio seguente, se a una variabile si assegna il nome new e quindi si prova il documento, il compilatore di Flash genera un errore:

```
// Questa parte di codice produce i risultati previsti
var helloStr:String = new String();
trace(helloStr.length); // 0
// Se invece si assegna a una variabile lo stesso nome di una classe
// incorporata
var new:String = "hello"; //Errore: previsto identificatore
var helloStr:String = new String();
trace(helloStr.length); // undefined
```

L'editor di ActionScript supporta i suggerimenti sul codice per le classi incorporate e per le variabili basate su tali classi. Se si desidera ricevere suggerimenti sul codice in Flash per un particolare tipo di oggetto assegnato a una variabile, è possibile eseguire una tipizzazione forte della variabile. I suggerimenti sul codice sono suggerimenti sulla sintassi che vengono visualizzati come descrizioni comandi e comprendono un menu a comparsa che aiuta a scrivere codice in modo più rapido.

Immettere ad esempio il seguente codice:

```
var members:Array = new Array();
members.
```

Quando viene digitato il punto (.) nel pannello Azioni, Flash visualizza l'elenco di metodi e proprietà disponibili per gli oggetti Array.

Per informazioni sulle convenzioni consigliate per l'assegnazione di nomi alle variabili, vedere [“Assegnazione di nomi alle variabili” a pagina 796](#).

Uso di variabili in un'applicazione

In questa sezione vengono utilizzate variabili in brevi frammenti di codice ActionScript. Prima di utilizzare una variabile in un'espressione, è necessario dichiararla e inizializzarla in uno script. Le espressioni sono combinazioni di operandi e operatori che rappresentano un valore. Nell'espressione `i+2`, ad esempio, `i` e `2` sono operandi e `+` è un operatore.

Se una variabile non viene inizializzata prima di essere utilizzata in un'espressione, tale variabile rimane non definita e può generare risultati imprevisti. Per ulteriori informazioni sulla scrittura di espressioni, consultare il [Capitolo 4, “Nozioni fondamentali sul linguaggio e la sintassi” a pagina 75](#).

Se si utilizza una variabile non definita, come nell'esempio seguente, in Flash Player 7 e nelle versioni successive la variabile acquisisce il valore `Nan` e lo script può generare risultati imprevisti:

```
var squared:Number = myNum * myNum;  
trace(squared); // Nan  
var myNum:Number = 6;
```

Nell'esempio seguente, l'istruzione che dichiara e inizializza la variabile `myNum` deve essere specificata per prima, affinché `squared` possa essere sostituito con un valore:

```
var myNum:Number = 6;  
var squared:Number = myNum * myNum;  
trace(squared); // 36
```

Un comportamento analogo si verifica quando si passa una variabile non definita a un metodo o a una funzione, come illustrato di seguito.

Per confrontare il passaggio di variabili definite e non definite a una funzione:

1. Trascinare un componente Button nello stage dal pannello Componenti.
2. Nella finestra di ispezione Proprietà, digitare **bad_button** nella casella di testo relativa al nome dell'istanza.
3. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
// Non funziona
function badClickListener(evt:Object):Void {
    getURL(targetUrl);
    var targetUrl:String = "http://www.macromedia.com";
}
bad_button.addEventListener("click", badClickListener);
```
4. Selezionando Controllo > Prova filmato, si può notare che il pulsante non funziona, vale a dire non apre la pagina Web.
5. Trascinare nello stage un altro componente Button. Selezionare il pulsante.
6. Nella finestra di ispezione Proprietà digitare **good_button** nella casella di testo relativa al nome dell'istanza.
7. Aggiungere il seguente codice ActionScript al fotogramma 1 della linea temporale, di seguito al codice già aggiunto:

```
// Funziona
function goodClickListener(evt:Object):Void {
    var targetUrl:String = "http://www.macromedia.com";
    getURL(targetUrl);
}
good_button.addEventListener("click", goodClickListener);
```

8. Selezionare Controllo > Prova filmato e fare clic sul secondo pulsante aggiunto allo stage.

Questo pulsante apre la pagina Web.

Il tipo di dati contenuto in una variabile determina come e quando il valore della variabile viene modificato. I tipi di dati di base, ad esempio le stringhe e i numeri, vengono *passati in base al valore*, ovvero viene utilizzato il valore corrente della variabile anziché un riferimento a tale valore. Esempi di tipi di dati complessi comprendono i tipi di dati Array e Object.

Nell'esempio seguente, `myNum` viene impostato su 15 e il valore viene copiato in `otherNum`. Quando si modifica `myNum` in 30 (nella riga 3 del codice), il valore di `otherNum` rimane 15 perché `otherNum` non cerca il proprio valore in `myNum`. La variabile `otherNum` contiene il valore di `myNum` ricevuto (nella riga 2 del codice).

Per utilizzare le variabili nel codice ActionScript:

1. Creare un nuovo documento Flash e salvarlo come **var_example.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var myNum:Number = 15;
var otherNum:Number = myNum;
myNum = 30;
trace(myNum); // 30
trace(otherNum); // 15
```

Quando si modifica `myNum` in 30 (nella riga 3 del codice), il valore di `otherNum` rimane 15 perché `otherNum` non cerca il proprio valore in `myNum`. La variabile `otherNum` contiene il valore di `myNum` ricevuto (nella riga 2 del codice).

3. Selezionare Controllo > Prova filmato per visualizzare i valori nel pannello Output.
4. Aggiungere ora il codice ActionScript seguente dopo il codice aggiunto al punto 2:

```
function sqr(myNum:Number):Number {
    myNum *= myNum;
    return myNum;
}
var inValue:Number = 3;
var outValue:Number = sqr(inValue);
trace(inValue); // 3
trace(outValue); // 9
```

Nel codice riportato, la variabile `inValue` contiene il valore di base 3. Il valore viene passato alla funzione `sqr()` e il valore restituito è 9. Il valore della variabile `inValue` non viene modificato anche se il valore di `myNum` nella funzione cambia.

5. Selezionare Controllo > Prova filmato per visualizzare i valori nel pannello Output.
- Poiché il tipo di dati Object può contenere un'elevata quantità di informazioni complesse, una variabile con questo tipo di dati non contiene un valore effettivo, ma soltanto un riferimento a un valore che può essere paragonato a un alias che punta al contenuto della variabile. Quando la variabile richiede il proprio valore, il riferimento recupera il contenuto e restituisce la risposta senza trasferire il valore alla variabile.

Per informazioni sul passaggio di una variabile per riferimento, vedere “[Passaggio di una variabile per riferimento](#)” a pagina 360.

Passaggio di una variabile per riferimento

Dato che i tipi di dati Array e Object contengono un riferimento a un valore e non il valore effettivo, prestare particolare attenzione quando si lavora con array e oggetti.

L'esempio seguente illustra come passare un oggetto in base al riferimento. Quando si crea una copia dell'array, si crea in realtà solo una copia del riferimento (o *alias*) al contenuto dell'array. Modificando il contenuto del secondo array, si modifica sia il contenuto del primo che del secondo array, perché entrambi puntano allo stesso valore.

Per passare un oggetto in base al riferimento:

1. Selezionare File > Nuovo e quindi Documento Flash per creare un nuovo file FLA e salvarlo come **copybyref.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var myArray:Array = new Array("tom", "josie");
var newArray:Array = myArray;
myArray[1] = "jack";
trace(myArray); // tom,jack
trace(newArray); // tom,jack
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

Questo codice ActionScript crea un oggetto Array denominato `myArray` che contiene due elementi. Viene creata la variabile `newArray` e viene passato un riferimento a `myArray`.

Quando il secondo elemento di `myArray` viene modificato in `jack`, l'operazione ha effetto su tutte le variabili che fanno riferimento a tale elemento. L'istruzione `trace()` invia `tom,jack` al pannello Output.

NOTA

Flash utilizza un indice con base zero, ovvero 0 è il primo elemento dell'array, 1 il secondo e così via.

Nell'esempio seguente, `myArray` contiene un oggetto Array, quindi l'array viene passato alla funzione `zeroArray()` in base al riferimento. La funzione `zeroArray()` accetta un oggetto Array come parametro e imposta tutti gli elementi di tale matrice su 0. Consente di modificare la matrice in quanto questa viene trasferita in base al riferimento.

Per passare un array in base al riferimento:

1. Selezionare File > Nuovo e quindi Documento Flash per creare un nuovo file FLA e salvarlo come **arraybyref.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
function zeroArray (theArr:Array):Void {  
    var i:Number;  
    for (i = 0; i < theArr.length; i++) {  
        theArr[i] = 0;  
    }  
}  
  
var myArr:Array = new Array();  
myArr[0] = 1;  
myArr[1] = 2;  
myArr[2] = 3;  
trace(myArr); // 1,2,3  
zeroArray(myArr);  
trace(myArr); // 0,0,0
```

3. Selezionare Controllo > Prova filmato per provare il codice ActionScript.

La prima istruzione `trace()` nel codice ActionScript visualizza il contenuto originale dell'array `myArray` (1,2,3). In seguito alla chiamata alla funzione `zeroArray()` e al passaggio di un riferimento all'array `myArray`, tutti i valori dell'array vengono sovrascritti e impostati a zero. L'istruzione `trace()` seguente visualizza il nuovo contenuto dell'array `myArray` (0,0,0). Dato che l'array viene passato in base al riferimento e non al valore, non occorre restituire il contenuto aggiornato dell'array dall'interno della funzione `zeroArray()`.

Per ulteriori informazioni sugli array, consultare il “[Informazioni sugli array](#)” a pagina 129.

Informazioni sulle variabili e l'area di validità

Per area di validità di una variabile si intende la porzione di codice in cui la variabile è nota (*definita*) e da cui può esservi fatto riferimento. L'area in cui la variabile è nota potrebbe essere una determinata linea temporale o una funzione; alcune variabili possono essere note globalmente in tutta l'applicazione. Per ulteriori informazioni sull'area di validità, vedere [“Informazioni su area di validità e identificazione” a pagina 86](#).

Comprendere il significato di area di validità delle variabili è importante se si sviluppano applicazioni Flash con ActionScript. Area di validità indica non solo quando e dove è possibile fare riferimento alle variabili, ma anche per quanto tempo una determinata variabile esiste nell'applicazione. Quando si definiscono variabili nel corpo di una funzione, queste cessano di esistere quando la funzione termina. Se si tenta di fare riferimento ad oggetti nell'area di validità scorretta o a variabili non più esistenti, nei documenti Flash vengono generati errori che causano comportamenti imprevisti o funzionalità mancanti.

In ActionScript esistono tre tipi di area di validità delle variabili:

- [Variabili globali](#) e funzioni globali: sono visibili per tutte le linee temporali e le aree di validità del documento. Le variabili globali sono pertanto definite in tutte le aeree del codice.
- [Variabili della linea temporale](#) sono disponibili per qualsiasi script della linea temporale specifica.
- [Variabili locali](#) sono disponibili nell'ambito del corpo della funzione in cui sono dichiarate (sono contraddistinte da parentesi graffe). Le variabili locali sono pertanto definite solo in una parte del codice.

Per linee guida sull'utilizzo dell'area di validità e delle variabili, vedere [Capitolo 4, “Informazioni su area di validità e identificazione” a pagina 86](#).

NOTA

Le classi di ActionScript 2.0 create dall'utente supportano aree di validità delle variabili pubbliche, private e statiche. Per ulteriori informazioni, vedere [“Informazioni sui membri di classe” a pagina 224](#) e [“Controllo dell'accesso dei membri delle classi” a pagina 248](#).

Non è possibile la tipizzazione forte delle variabili globali. Per informazioni e per una soluzione alternativa, vedere [“Variabili globali” a pagina 363](#).

Variabili globali

Le variabili e le funzioni globali sono visibili per tutte le linee temporali e le aree di validità del documento. Per dichiarare (o *creare*) una variabile con un'area di validità globale, utilizzare l'identificatore `_global` prima del nome della variabile senza utilizzare la sintassi `var =`. Nel codice seguente viene creata ad esempio la variabile globale `myName`:

```
var _global.myName = "George"; // Sintassi scorretta per la variabile  
    globale  
_global.myName = "George"; // Sintassi corretta per la variabile globale
```

Se tuttavia una variabile locale viene inizializzata con lo stesso nome di una variabile globale, non è possibile accedere alla variabile globale mentre ci si trova nell'area di validità di quella locale, come mostrato nell'esempio seguente:

```
_global.counter = 100; // Dichiara la variabile globale  
trace(counter); // Accede alla variabile globale e visualizza 100  
function count():Void {  
    for (var counter:Number = 0; counter <= 2; counter++) { // Variabile  
        locale  
        trace(counter); // Accede alla variabile locale e visualizza da 0 a 2  
    }  
}  
count();  
trace(counter); // Accede alla variabile globale e visualizza 100
```

Nell'esempio è dimostrato che non è possibile accedere alla variabile globale nell'area di validità della funzione `count()`. È possibile tuttavia accedere alla variabile con area di validità globale utilizzando con la variabile il prefisso `_global`, ad esempio utilizzando con il contatore il prefisso `_global` nel codice seguente:

```
trace(_global.counter);
```

È impossibile assegnare la tipizzazione forte dei dati alle variabili che vengono create nell'area di validità `_global`, perché è necessario usare la parola chiave `var` quando si assegna un tipo di dati. Ad esempio, non è possibile eseguire l'operazione indicata di seguito:

```
_global.foo:String = "foo"; //errore di sintassi  
var _global.foo:String = "foo"; //errore di sintassi
```

La funzione di sicurezza sandbox di Flash Player 7 e delle versioni successive applica restrizioni in caso di accesso a variabili globali da file SWF caricati da domini di sicurezza separati. Per ulteriori informazioni, vedere [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)
[a pagina 733](#).

Variabili della linea temporale

Le variabili della linea temporale sono disponibili per qualsiasi script di tale linea temporale. Per dichiarare le variabili della linea temporale, utilizzare l'istruzione `var` e inizializzare le variabili in qualsiasi fotogramma della linea temporale. La variabile è disponibile per quel fotogramma e per tutti i fotogrammi successivi, come nell'esempio che segue.

Per utilizzare variabili della linea temporale in un documento:

1. Creare un nuovo documento Flash e denominarlo **timelinevar.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var myNum:Number = 15; /* Inizializzata nel fotogramma 1, pertanto  
disponibile in tutti i fotogrammi */
```
3. Selezionare il fotogramma 20 della linea temporale.
4. Selezionare Inserisci > Linea temporale > Fotogramma chiave vuoto.
5. Con il nuovo fotogramma chiave selezionato, digitare il codice ActionScript seguente nel pannello Azioni:

```
trace(myNum);
```
6. Selezionare Controllo > Prova filmato per provare il nuovo documento.

Il valore 15 viene visualizzato nel pannello Output dopo circa un secondo. Dato che i documenti Flash sono ciclici per impostazione predefinita, il valore 15 viene analizzato in continuazione nel pannello Output ogni volta che l'indicatore di riproduzione raggiunge il fotogramma 20 nella linea temporale. Per arrestare la ripetizione ciclica, aggiungere `stop();` dopo l'istruzione `trace()`.

Dichiarare una variabile della linea temporale prima di tentare di accedervi in uno script. Se, ad esempio, si inserisce il codice `var myNum:Number = 15;` nel fotogramma 20, gli script associati ai fotogrammi precedenti il 20 non possono accedere a `myNum` e risultano non definiti anziché contenere il valore 15.

Variabili locali

Quando si utilizza l'istruzione `var` all'interno di un blocco di funzione, si dichiarano *variabili locali*. Una variabile locale dichiarata all'interno di un blocco di funzione (detto anche *definizione di funzione*) è definita all'interno dell'area di validità del blocco di funzione. La sua validità termina alla fine del blocco di funzione. La variabile locale esiste pertanto solo all'interno della funzione.

Se, ad esempio, si dichiara una variabile denominata `myStr` all'interno di una funzione denominata `localScope`, tale variabile non è disponibile all'esterno della funzione.

```
function localScope():Void {  
    var myStr:String = "local";  
}  
localScope();  
trace(myStr); // Non definita perché myStr non è definita a livello globale
```

Se la variabile locale è stata dichiarata con lo stesso nome di una variabile della linea temporale, la definizione locale ha la precedenza sulla definizione della linea temporale all'interno dell'area di validità della variabile locale. La variabile della linea temporale è ancora valida all'esterno della funzione. Il codice seguente, ad esempio, crea una variabile di tipo `String` della linea temporale denominata `str1`, quindi una variabile locale con lo stesso nome all'interno della funzione `scopeTest()`. L'istruzione `trace` all'interno della funzione genera la definizione locale della variabile, ma all'esterno della funzione genera la definizione della variabile della linea temporale.

```
var str1:String = "Timeline";  
function scopeTest():Void {  
    var str1:String = "Local";  
    trace(str1); // Variabile locale  
}  
scopeTest();  
trace(str1); // Linea temporale
```

Nell'esempio successivo viene illustrato come alcune variabili sono valide solo per la durata di una determinata funzione e possono generare errori se si tenta di farvi riferimento all'esterno dell'area di validità della funzione.

Per utilizzare variabili locali in un'applicazione:

1. Creare un nuovo documento Flash.
2. Aprire il pannello Azioni (Finestra > Azioni) e aggiungere il seguente codice ActionScript nel fotogramma 1 della linea temporale:

```
function sayHello(nameStr:String):Void {  
    var greetingStr:String = "Hello, " + nameStr;  
    trace(greetingStr);  
}  
sayHello("world"); // Hello, world  
trace(nameStr); // undefined  
trace(greetingStr); // undefined
```
3. Selezionare Controllo > Prova filmato per provare il documento.

Flash visualizza la stringa "Hello, world" nel pannello Output e undefined per i valori di nameStr e greetingStr perché le variabili non sono più disponibili nell'area di validità corrente. È possibile fare riferimento a nameStr e greetingStr solo durante l'esecuzione della funzione sayHello. All'uscita dalla funzione la variabile non è più valida.

Le variabili i e j sono spesso utilizzate come contatori di ciclo. Nell'esempio seguente, la variabile i è utilizzata come variabile locale ed esiste esclusivamente all'interno della funzione initArray():

```
var myArr:Array = new Array();  
function initArray(arrayLength:Number):Void {  
    var i:Number;  
    for(i = 0; i < arrayLength; i++) {  
        myArr[i] = i + 1;  
    }  
    trace(myArr); // <vuota>  
    initArray(3);  
    trace(myArr); // 1,2,3  
    trace(i); // undefined
```



La sintassi seguente viene inoltre utilizzata comunemente per un ciclo for: for (var i:Number = 0; i < arrayLength; i++) {...}.

Questo esempio visualizza undefined nell'ambiente di prova di Flash perché la variabile i non è definita nella linea temporale principale, ma è valida solo nella funzione initArray().

Le variabili locali possono inoltre contribuire a prevenire conflitti tra nomi e quindi risultati imprevisti nell'applicazione. Se, ad esempio, si utilizza age come variabile locale, è possibile utilizzarla per memorizzare l'età di una persona in un contesto e l'età del figlio di quella persona in un altro contesto. Poiché tali variabili vengono utilizzate in aree di validità separate, non si verificano conflitti.

È sempre consigliabile utilizzare variabili locali nel corpo di una funzione, in modo che la funzione possa essere eseguita come entità di codice indipendente. Una variabile locale è modificabile soltanto all'interno del blocco di codice di appartenenza. Se un'espressione di una funzione utilizza una variabile globale, è possibile che il codice o gli eventi esterni alla funzione modifichino il valore della variabile, con conseguente modifica della funzione stessa.

È possibile assegnare un tipo di dati a una variabile locale al momento della dichiarazione. In questo modo si evita di assegnare tipi di dati scorretti alle variabili esistenti. Per ulteriori informazioni, vedere “[Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati](#)” a pagina 344.

Informazioni sul caricamento di variabili

Nelle sezioni seguenti vengono caricate variabili dal server in modi diversi o in un documento da una stringa URL o FlashVars nel codice HTML. FlashVars può essere utilizzata per passare variabili in Flash. Le variabili possono essere utilizzate all'esterno di un file SWF in diversi modi.

Per ulteriori informazioni sul caricamento di variabili (ad esempio coppie nome/valore) consultare il [Capitolo 16, “Operazioni con i dati esterni” a pagina 689](#).

In un file SWF le variabili possono essere utilizzate in modi diversi in base allo scopo che ci si prefigge. Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Uso delle variabili dall'URL](#)” a pagina 367
- “[Uso di FlashVars in un'applicazione](#)” a pagina 370
- “[Caricamento di variabili da un server](#)” a pagina 372

Uso delle variabili dall'URL

Nelle applicazioni o in esempi semplici di Flash potrebbe essere necessario passare valori da una pagina HTML al documento Flash. I valori passati sono noti anche come la *stringa di query*, o *variabili con codifica URL*. Le variabili URL sono utili ad esempio per creare un menu in Flash. È possibile inizializzare il menu per indicare la navigazione corretta per impostazione predefinita oppure generare un visualizzatore di immagini in Flash e definire un'immagine predefinita da visualizzare sul sito Web.

Per utilizzare variabili URL in un documento:

1. Creare un documento Flash e denominarlo **urlvariables.fla**.
2. Selezionare File > Salva con nome e salvare il documento sul desktop.
3. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
myTxt.text = _level0.myURL;
```

4. Selezionare Controllo > Prova filmato per provare il file SWF in Flash Player.

Nel campo di testo viene visualizzato `undefined`. Per garantire che le variabili siano state definite correttamente prima di continuare, controllare l'esistenza delle variabili in Flash verificando se risultano non definite.

5. Per eseguire questa verifica, modificare il codice ActionScript aggiunto al pannello Azioni al punto 3 in base al codice seguente, aggiungendo il codice riportato in **grassetto**:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
    myTxt.text = "myURL is not defined";
} else {
    myTxt.text = _level0.myURL;
}
```

Alla pubblicazione del documento Flash, viene creato un documento HTML per impostazione predefinita nella stessa directory del file SWF. Se non viene creato, selezionare File > Impostazioni pubblicazione e quindi HTML nella scheda Formati.

Ripubblicare il documento.

Il codice seguente dimostra l'HTML del documento utilizzato per incorporare un documento Flash in una pagina HTML e permette di comprendere il funzionamento delle variabili URL nel seguente passaggio (in cui viene aggiunto codice aggiuntivo per le variabili URL).

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
        codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
        swflash.cab#version=8,0,0,0" width="550" height="400"
        id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" bgcolor="#ffffff"
        width="550" height="400" name="urlvariables" align="middle"
        allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
        pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

- 6.** Per passare variabili dal documento HTML generato al documento Flash, è possibile aggiungere le variabili dopo il percorso e il nome di file (urlvariables.swf). Aggiungere il **testo in grassetto** al file HTML generato sul desktop.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
       codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
       swflash.cab#version=8,0,0,0" width="550" height="400"
       id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf?myURL=http://
       weblogs.macromedia.com" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf?myURL=http://weblogs.macromedia.com"
       quality="high" bgcolor="#ffffff" width="550" height="400"
       name="urlvariables" align="middle" allowScriptAccess="sameDomain"
       type="application/x-shockwave-flash" pluginspage="http://
       www.macromedia.com/go/getflashplayer" />
</object>
```

- 7.** Per passare a Flash più variabili, separare le coppie nome/valore con una e commerciale (&). Cercare il codice seguente aggiunto al punto 6:

?myURL=http://weblogs.macromedia.com

Sostituirlo con il testo seguente:

?myURL=http://weblogs.macromedia.com&myTitle=Macromedia+News+Aggregator

Tenere a mente che occorre apportare le stesse modifiche sia al tag `object` che al tag `embed` per garantire la coerenza tra tutti i browser. Come si può notare, le parole sono separate da segni +, perché i valori hanno codifica URL e il segno + rappresenta uno spazio vuoto singolo.



Per un elenco dei caratteri speciali con codifica URL più utilizzati, vedere la nota tecnica Flash, [Codifica URL: lettura di caratteri speciali da un file di testo](#).

Dato che la e commerciale (&) funge da delimitatore per diverse coppie nome/valore, se i valori che vengono passati contengono e commerciali, potrebbero verificarsi risultati imprevisti. Data la natura delle coppie nome/valore e l'analisi, se i seguenti valori vengono passati a Flash:

my.swf?name=Ben++Jerry&flavor=Half+Baked

Flash genera le seguenti variabili (e i seguenti valori) nell'area di validità principale:

```
'name': 'Ben ' (note space at end of value)
'Jerry': '' (note space at beginning of variable name and an empty
             value)
'flavor': 'Half Baked'
```

Per evitare questo comportamento, *sostituire* il carattere della e commerciale (&) nella coppia nome/valore con l'equivalente nella codifica URL (%26).

- 8.** Aprire il documento urlvariables.html e cercare il codice seguente:

```
?myURL=http://weblogs.macromedia.com&myTitle=Macromedia+News+Aggregator
```

Sostituirlo con il codice seguente:

```
?myURL=Ben%26Jerry&flavor=Half+Baked
```

- 9.** Salvare l'HTML corretto e provare nuovamente il documento Flash.

Flash crea le seguenti coppie nome/valore.

```
'name': 'Ben & Jerry'  
'flavor': 'Half Baked'
```



Tutti i browser supportano stringhe di lunghezza fino a 65535 byte. FlashVars deve essere assegnata sia nel tag object che nel tag embed affinché il documento possa funzionare con tutti i browser.

Uso di FlashVars in un'applicazione

L'uso di FlashVars per passare variabili a Flash è analogo al passaggio di variabili tramite l'URL nel codice HTML. Con FlashVars, anziché essere passate dopo il nome del file, le variabili vengono passate in un tag param separato e nel tag embed.

Per utilizzare FlashVars in un documento:

1. Creare un nuovo documento Flash e denominarlo **myflashvars.fla**.
2. Selezionare File > Impostazioni pubblicazione e verificare che sia selezionato HTML, quindi fare clic su OK per chiudere la finestra di dialogo.
3. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);  
myTxt.autoSize = "left";  
if (_level0.myURL == undefined) {  
    myTxt.text = "myURL is not defined";  
} else {  
    myTxt.text = _level0.myURL;  
}
```



Per impostazione predefinita, il codice HTML viene pubblicato nella stessa posizione di myflashvars.fla.

4. Selezionare File > Pubblica per pubblicare i file SWF e HTML.
5. Aprire la directory contenente i file pubblicati, dove è stato salvato myflashvars.fla, e il documento HTML (per impostazione predefinita myflashvars.html) in un editor HTML come Dreamweaver o Blocco note.

- 6.** Aggiungere il codice riportato in **grassetto** di seguito, affinché il documento HTML sia analogo a quanto segue:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
       codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
       swflash.cab#version=8,0,0,0" width="550" height="400" id="myflashvars"
       align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="myflashvars.swf" />
<param name="FlashVars" value="myURL=http://weblogs.macromedia.com/">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="myflashvars.swf" FlashVars="myURL=http://
       weblogs.macromedia.com/" quality="high" bgcolor="#ffffff" width="550"
       height="400" name="myflashvars" align="middle"
       allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
       pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Questo codice passa una sola variabile denominata `myURL` che contiene la stringa `http://weblogs.macromedia.com`. Quando il file SWF viene caricato, nell'area di validità `_level0` viene creata una proprietà denominata `myURL`. Uno dei vantaggi derivanti dall'uso di `FlashVars` o dal passaggio di variabili tramite l'URL è rappresentato dal fatto che le variabili sono disponibili immediatamente in Flash quando il file SWF viene caricato. Non è pertanto necessario scrivere funzioni per verificare se le variabili sono state caricate completamente, come invece si dovrebbe fare se le variabili vengono caricate con `LoadVars` o XML.

- 7.** Salvare le modifiche al documento HTML e chiuderlo.
8. Fare doppio clic su `myflashvars.html` per provare l'applicazione.

Nel file SWF viene visualizzato il testo `http://weblogs.macromedia.com`, una variabile del file HTML.

NOTA

Tutti i browser supportano stringhe di lunghezza fino a 65.535 byte. `FlashVars` deve essere assegnata sia nel tag `object` che nel tag `embed` affinché il documento possa funzionare con tutti i browser.

Caricamento di variabili da un server

Le variabili possono essere caricate in Flash da origini esterne, ad esempio file di testo, documenti XML e così via, in modi diversi. Per ulteriori informazioni sul caricamento di variabili (tra cui coppie nome/valore) consultare il [Capitolo 16, “Operazioni con i dati esterni” a pagina 689](#).

In Flash le variabili possono essere caricate facilmente tramite la classe LoadVars, come nell'esempio che segue.

Per caricare variabili da un server:

1. Creare un nuovo documento Flash.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean):Void {
    if (success) {
        trace(this.dayNames); // Sunday,Monday,Tuesday,...
    } else {
        trace("Error");
    }
}
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

Il codice carica un file di testo da un server remoto e ne analizza le coppie nome/valore.

SUGGERIMENTO

Scaricare o visualizzare il file di testo (<http://www.helpexamples.com/flash/params.txt>) in un browser per sapere come sono formattate le variabili.

3. Selezionare Controllo > Prova filmato per provare il documento.

Se il file viene caricato completamente, viene chiamato l'evento `complete` e nel pannello Output viene visualizzato il valore di `dayNames`. Se il file di testo non può essere scaricato, l'argomento `success` viene impostato su `false` e nel pannello Output viene visualizzato il testo `Error`.

Uso di variabili in un progetto

Quando si creano animazioni o applicazioni con Flash a volte può non essere necessario utilizzare variabili nel progetto. Se ad esempio si crea un sistema di login, potrebbe essere necessario utilizzare variabili per determinare se il nome utente e la password sono validi o se sono stati immessi.

Per ulteriori informazioni sul caricamento di variabili (ad esempio coppie nome/valore) consultare il [Capitolo 16, “Operazioni con i dati esterni” a pagina 689](#).

Nell'esempio seguente, le variabili vengono utilizzate per memorizzare il percorso di un'immagine caricata con la classe Loader, una variabile per l'istanza della classe Loader e un paio di funzioni chiamate in base all'esito di caricamento del file.

Per utilizzare variabili in un progetto:

1. Creare un nuovo documento Flash e salvarlo come **imgloader.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```
/* Specifica l'immagine predefinita se non è stato passato alcun valore
   con FlashVars. */
var imgUrl:String = "http://www.helpexamples.com/flash/images/
   image1.jpg";
if (_level0.imgURL != undefined) {
   // Se è stata specificata un'immagine, sovrascrive il valore
   predefinito.
   imgUrl = _level0.imgURL;
}

this.createEmptyMovieClip("img_mc", 10);
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip):Void {
   target_mc._x = (Stage.width - target_mc._width) / 2;
   target_mc._y = (Stage.height - target_mc._height) / 2;
}
mcListener.onLoadError = function(target_mc:MovieClip):Void {
   target_mc.createTextField("error_txt", 1, 0, 0, 100, 20);
   target_mc.error_txt.autoSize = "left";
   target_mc.error_txt.text = "Error downloading specified image;\n\t" +
   target_mc._url;
}
var myMCL:MovieClipLoader = new MovieClipLoader();
myMCL.addListener(mcListener);
myMCL.loadClip(imgUrl, img_mc);
```

La prima riga di codice specifica l'immagine che si desidera caricare in modo dinamico nel documento Flash. Di seguito si controlla se è stato specificato un nuovo valore per `imgURL` utilizzando FlashVars o variabili con codifica URL. In caso positivo, l'URL dell'immagine predefinito viene sovrascritto con il nuovo valore. Per informazioni sull'uso di variabili URL, vedere “[Uso delle variabili dall'URL](#)” a pagina 367. Per informazioni su FlashVars, vedere “[Uso di FlashVars in un'applicazione](#)” a pagina 370.

Le due righe di codice successive definiscono l'istanza MovieClip e un oggetto Listener per l'istanza futura MovieClipLoader. L'oggetto Listener di MovieClipLoader definisce due gestori di eventi, ovvero `onLoadInit` e `onLoadError` che vengono richiamati quando l'immagine viene caricata e inizializzata sullo stage oppure se si verifica un errore durante il caricamento. Viene quindi creata un'istanza MovieClipLoader e utilizzato il metodo `addListener()` per aggiungere l'oggetto listener definito in precedenza a MovieClipLoader. Infine l'immagine viene caricata e attivata quando si chiama il metodo `MovieClipLoader.loadClip()` che specifica il file dell'immagine da caricare e il clip filmato di destinazione in cui eseguire il caricamento.

3. Selezionare Controllo > Prova filmato per provare il documento.

Dato che il documento Flash viene provato nello strumento di creazione, non viene passato alcun valore per `imgUrl` da FlashVars o nell'URL e quindi viene visualizzata l'immagine predefinita.

4. Salvare il documento Flash e selezionare File > Pubblica per pubblicare il file come documento SWF e HTML.



Nella finestra di dialogo Impostazioni pubblicazione devono essere selezionati sia Flash che HTML. Selezionare File > Impostazioni pubblicazione e fare clic sulla scheda Formati. Quindi, selezionare entrambe le opzioni.

5. Se il documento viene provato nello strumento apposito (selezionando Controllo > Prova filmato) o in un browser locale (File > Anteprima pubblicazione > HTML), l'immagine viene centrata sia verticalmente che orizzontalmente sullo stage.

- 6.** Modificare il documento HTML generato in un editor, ad esempio Dreamweaver o Blocco note e modificare l'HTML come segue:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
       codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
       swflash.cab#version=8,0,0,0" width="550" height="400"
       id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="FlashVars" value="imgURL=http://www.helpexamples.com/flash/
       images/image2.jpg">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" FlashVars="imgURL=http://
       www.helpexamples.com/flash/images/image2.jpg" bgcolor="#ffffff"
       width="550" height="400" name="urlvariables" align="middle"
       allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
       pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

- 7.** Provare il documento HTML per visualizzare le modifiche. Nel file SWF viene visualizzata un'immagine specificata nel codice HTML.

Per modificare questo esempio in modo da usare le proprie immagini, è necessario modificare il valore FlashVars (la stringa racchiusa tra virgolette).

Organizzazione di dati in oggetti

Probabilmente si ha familiarità con il posizionamento di oggetti sullo stage. Nello stage, ad esempio, può essere presente un oggetto MovieClip che a sua volta contiene altri clip filmato. Campi di testo, clip filmato e pulsanti vengono spesso denominati oggetti quando vengono posizionati sullo stage.

Gli oggetti in ActionScript sono insiemi di proprietà e metodi. Ogni oggetto ha un proprio nome ed è un'istanza di una classe. Gli oggetti incorporati derivano da classi predefinite di ActionScript. La classe incorporata Date, ad esempio, fornisce informazioni provenienti dall'orologio di sistema del computer dell'utente. La classe incorporata LoadVars può essere utilizzata per caricare variabili nel file SWF.

Tramite ActionScript è inoltre possibile creare oggetti e classi. Un oggetto può essere creato per contenere un insieme di dati, ad esempio il nome, l'indirizzo e il numero di telefono di una persona, oppure informazioni sui colori di un'immagine. L'organizzazione dei dati in oggetti garantisce una migliore organizzazione dei documenti Flash. Per informazioni generali sulla creazione di una classe personalizzata che contenga un insieme di metodi e proprietà, vedere [“Creazione di file di classi personalizzate” a pagina 208](#). Per informazioni dettagliate sulle classi incorporate e personalizzate, consultare il [Capitolo 6, “Classi” a pagina 197](#).

ActionScript fornisce vari modi per creare un oggetto. L'esempio seguente crea oggetti semplici in due modi diversi, quindi esegue elaborazioni cicliche del contenuto di tali oggetti.

Per creare oggetti semplici in Flash:

1. Creare un nuovo documento Flash e salvarlo come **simpleObjects.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
// Primo modo
var firstObj:Object = new Object();
firstObj.firstVar = "hello world";
firstObj.secondVar = 28;
firstObj.thirdVar = new Date(1980, 0, 1); // January 1, 1980
```

Questo codice, che rappresenta una delle modalità di creazione di un oggetto semplice, crea una nuova istanza di un oggetto e definisce alcune proprietà al suo interno.

3. Immettere ora il codice ActionScript seguente dopo il codice aggiunto al punto 2:

```
// Secondo modo
var secondObj:Object = {firstVar:"hello world", secondVar:28,
    thirdVar:new Date(1980, 0, 1)};
```

Questa rappresenta un'ulteriore modalità di creazione di un oggetto. I due oggetti sono equivalenti. Il codice appena riportato crea un nuovo oggetto e inizializza alcune proprietà utilizzando la notazione abbreviata dell'oggetto.

4. Per eseguire cicli su ognuno degli oggetti precedenti e visualizzarne il contenuto, aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale dopo il codice appena immesso:

```
var i:String;
for (i in firstObj) {
    trace(i + ": " + firstObj[i]);
}
```

5. Selezionare Controllo > Prova filmato. Viene visualizzato il testo seguente nel pannello Output:

```
firstVar: hello world
secondVar: 28
thirdVar: Tue Jan 1 00:00:00 GMT-0800 1980
```

Per creare oggetti è anche possibile utilizzare array. Anziché definire una serie di variabili come `firstname1`, `firstname2` e `firstname3` per rappresentare un insieme di variabili, è possibile creare un array di oggetti per rappresentare gli stessi dati, come dimostrato di seguito.

Per utilizzare un array per creare un oggetto:

1. Creare un nuovo documento Flash e salvarlo come **arrayObject.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice ActionScript seguente:

```
var usersArr:Array = new Array();
usersArr.push({firstname:"George"});
usersArr.push({firstname:"John"});
usersArr.push({firstname:"Thomas"});
```

L'organizzazione di variabili in array e oggetti facilita l'elaborazione ciclica delle variabili e la visualizzazione dei relativi valori, come illustrato al punto seguente.

3. Immettere il codice seguente dopo il codice ActionScript aggiunto al punto 2:

```
var i:Number;
for (i = 0; i < usersArr.length; i++) {
    trace(usersArr[i].firstname); // George, John, Thomas
}
```

4. Selezionare Controllo > Prova filmato. Viene visualizzato il testo seguente nel pannello Output:

```
George
John
Thomas
```

Anche l'esempio seguente consente di eseguire elaborazioni cicliche di oggetti, ma è scritto in modo diverso. In questo esempio viene creato un oggetto che viene elaborato in modo ciclico tramite un ciclo `for..in` e ogni proprietà viene visualizzata nel pannello Output.

```
var myObj:Object = {var1:"One", var2:"Two", var3:18, var4:1987};
var i:String;
for (i in myObj) {
    trace(i + ": " + myObj[i]);
}
//Fornisce il seguente output:
/*
    var1: uno
    var2: due
    var3: 18
    var4: 1987
*/
```

Per informazioni sulla creazione di cicli, consultare il [Capitolo 4, “Uso dei cicli for” a pagina 123](#). Per informazioni sui cicli `for..in`, consultare il [“Uso dei cicli for..in” a pagina 125](#). Per ulteriori informazioni sugli oggetti, vedere [Capitolo 6, “Classi” a pagina 197](#).

Informazioni sull'inserimento

ActionScript 2.0 consente di inserire un tipo di dati in un altro. Inserire un oggetto in un tipo diverso significa convertire il valore memorizzato dall'oggetto o dalla variabile in un tipo diverso.

I risultati di un inserimento di tipo sono diversi in base ai tipi di dati. Per inserire un oggetto in un tipo diverso, racchiudere il nome dell'oggetto tra parentesi () e farlo precedere dal nome del nuovo tipo. Il codice seguente, ad esempio accetta un valore booleano e lo inserisce in un numero intero.

```
var myBoolean:Boolean = true;  
var myNumber:Number = Number(myBoolean);
```

Per ulteriori informazioni sull'inserimento, consultare i seguenti argomenti:

- [“Informazioni sull'inserimento di oggetti” a pagina 378](#)

Informazioni sull'inserimento di oggetti

La sintassi per l'inserimento è `tipo(elemento)` e può essere utilizzata nel punto in cui il compilatore deve trattare l'elemento come se fosse del tipo di dati `tipo`. L'inserimento è fondamentalmente una chiamata di funzione che restituisce `null` se l'operazione non ha esito positivo in fase di runtime (nei file pubblicati per Flash Player 7 o versioni successive; i file pubblicati per Flash Player 6 non dispongono di supporto in fase di runtime per gli inserimenti non riusciti). Se l'inserimento riesce, la chiamata di funzione restituisce l'oggetto originale. Il compilatore tuttavia non è in grado di determinare che un inserimento avrà esito negativo in fase di runtime e in tali casi non genera errori in fase di compilazione.

Nel codice seguente viene presentato un esempio:

```
// Both the Cat and Dog classes are subclasses of the Animal class  
function bark(myAnimal:Animal) {  
    var foo:Dog = Dog(myAnimal);  
    foo.bark();  
}  
var curAnimal:Animal = new Dog();  
bark(curAnimal); // Funziona  
curAnimal = new Cat();  
bark(curAnimal); // Non funziona
```

In questo esempio si asserisce che `foo` è un oggetto `Dog` e pertanto il compilatore presume che `foo.bark()` sia un'istruzione valida. Il compilatore non è tuttavia in grado di prevedere che l'inserimento non avrà esito positivo, ovvero che si è tentato di inserire un oggetto `Cat` in un tipo `Animal`, e pertanto non genera alcun errore in fase di compilazione. Se nello script si verifica l'esito positivo dell'inserimento, è possibile rilevare errori di inserimento in fase di runtime, come nell'esempio che segue.

```
function bark(myAnimal:Animal) {  
    var foo:Dog = Dog(myAnimal);  
    if (foo) {  
        foo.bark();  
    }  
}
```

È possibile inserire un'espressione in un'interfaccia. Se l'espressione è un oggetto che implementa l'interfaccia oppure dispone di una classe base che implementa l'interfaccia, l'inserimento ha esito positivo. In caso contrario, ha esito negativo.

NOTA

L'inserimento in un tipo null o undefined restituisce `undefined`.

Non è possibile sostituire i tipi di dati di base che dispongono di una funzione di conversione globale corrispondente con un operatore di inserimento con lo stesso nome, perché le funzioni di conversione globali hanno la priorità rispetto agli operatori di inserimento. È possibile, ad esempio, eseguire l'inserimento in un tipo `Array` perché la funzione di conversione `Array()` ha la priorità sull'operatore di inserimento.

Questo esempio definisce due variabili di tipo `String` (`firstNum` e `secondNum`) che vengono sommate. Inizialmente i numeri vengono concatenati e non sommati perché i valori sono di tipo `String`. La seconda istruzione `trace` converte entrambi i numeri in un tipo di dati `Number` prima di eseguire la somma che consente di ottenere il risultato corretto. La conversione dei dati è importante se si lavora con dati caricati tramite XML o FlashVars, come nell'esempio che segue:

```
var firstNum:String = "17";  
var secondNum:String = "29";  
trace(firstNum + secondNum); // 1729  
trace(Number(firstNum) + Number(secondNum)); // 46
```

Per ulteriori informazioni sulle funzioni di conversione dei dati, vedere la voce relativa a ogni funzione di conversione nella *Guida di riferimento di ActionScript 2.0: Array function, Boolean function, Number function, Object function e String function*.

Operazioni con i clip filmato

I clip filmato sono come file SWF autonomi che vengono eseguiti indipendentemente l'uno dall'altro e dalla linea temporale in cui sono contenuti. Ad esempio, se la linea temporale principale presenta un solo fotogramma e un clip filmato ne presenta dieci, ciascun fotogramma contenuto nel clip filmato viene riprodotto durante la riproduzione del file SWF principale. Un clip filmato può a sua volta contenere altri clip filmato o *filmati nidificati*. I clip filmato nidificati sono organizzati in base a relazioni gerarchiche, ovvero il *clip principale* contiene uno o più *clip secondari*.

È possibile denominare istanze di clip filmato per poterle identificare in modo univoco come oggetti controllabili tramite ActionScript. Quando a un'istanza di clip filmato viene assegnato un *nome di istanza*, tale nome identifica il clip filmato come oggetto della classe MovieClip. Utilizzare le proprietà e i metodi della classe MovieClip per controllare l'aspetto visivo e il comportamento dei clip filmato in fase di runtime.

I clip filmato possono essere considerati oggetti autonomi che rispondono a eventi, inviano messaggi ad altri oggetti clip filmato, aggiornano il proprio stato e gestiscono i clip secondari. In questo modo, i clip filmato costituiscono le fondamenta di una *architettura basata su componenti* in Macromedia Flash Basic 8 e Macromedia Flash Professional 8. Infatti, i componenti disponibili nel pannello Componenti (Finestra > Componenti) sono clip filmato sofisticati, progettati e programmati per presentare un aspetto visivo e comportamenti particolari.

Per informazioni sull'uso dell'API di disegno (metodi di disegno della classe MovieClip), su filtri, fusione, animazione con script e altro ancora, consultare il [Capitolo 13, “Animazione, filtri e disegni”](#)

Per ulteriori informazioni sui clip filmato, consultare i seguenti argomenti:	
Informazioni sul controllo di clip filmato in ActionScript382
Chiamata di più metodi su un singolo clip filmato384
Caricamento e scaricamento di file SWF385
Modifica della posizione e dell'aspetto di un clip filmato388
Trascinamento di clip filmato390
Creazione di clip filmato in fase di runtime391
Aggiunta di parametri a clip filmato creati dinamicamente396
Gestione delle profondità nei clip filmato398
Informazioni sulla memorizzazione nella cache e sullo scorrimento dei clip filmato in ActionScript401
Uso di clip filmato come maschere409
Gestione di eventi clip filmato411
Assegnazione di una classe a un simbolo clip filmato412
Inizializzazione delle proprietà delle classi413

Informazioni sul controllo di clip filmato in ActionScript

È possibile utilizzare le funzioni globali di ActionScript o i metodi della classe MovieClip per eseguire operazioni su un clip filmato. Alcuni metodi della classe MovieClip eseguono le stesse operazioni delle funzioni con lo stesso nome; altri metodi MovieClip, quali `hitTest()` e `swapDepths()` non presentano funzioni corrispondenti con lo stesso nome.

Nell'esempio seguente viene illustrata la differenza tra l'uso di un metodo e di una funzione. Ogni istruzione duplica l'istanza `my_mc`, assegna un nome al nuovo clip `newClip` e lo posiziona a una profondità di 5.

```
my_mc.duplicateMovieClip("new_mc", 5);
duplicateMovieClip(my_mc, "new_mc", 5);
```

Se sia una funzione che un metodo offrono comportamenti simili, è possibile scegliere se controllare i clip filmato tramite la funzione o il metodo. La scelta dipende dalle preferenze e dall'abilità dello sviluppatore nel creare script con ActionScript. Indipendentemente dall'uso di una funzione o di un metodo, la linea temporale target deve essere caricata in Flash Player quando viene chiamata la funzione o il metodo.

Per utilizzare un metodo, chiamarlo attraverso il percorso target del nome dell'istanza, seguito da un punto, dal nome del metodo e dai parametri, come nelle istruzioni seguenti:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

Nella prima istruzione, `play()` sposta l'indicatore di riproduzione nell'istanza `myMovieClip`. Nella seconda istruzione il metodo `gotoAndPlay()` porta l'indicatore di riproduzione di `childClip` (un elemento secondario dell'istanza `parentClip`) sul fotogramma 3 e continua a spostare l'indicatore di riproduzione.

Le funzioni globali che controllano una linea temporale dispongono di un parametro `target` che consente di specificare il percorso target dell'istanza che si desidera controllare. Nello script seguente, ad esempio, `startDrag()` è associato all'istanza su cui è presente il codice e la rende trascinabile:

```
my_mc.onPress = function() {
    startDrag(this);
};

my_mc.onRelease = function() {
    stopDrag();
};
```

Le funzioni seguenti sono associate a clip filmato: `loadMovie()`, `unloadMovie()`, `loadVariables()`, `setProperty()`, `startDrag()`, `duplicateMovieClip()` e `removeMovieClip()`. Per usare queste funzioni, immettere un percorso target per il parametro `target` della funzione che consente di indicare il target della funzione.

I seguenti metodi MovieClip sono in grado di controllare i clip filmato o i livelli caricati e non hanno funzioni corrispondenti: `MovieClip.attachMovie()`, `MovieClip.createEmptyMovieClip()`, `MovieClip.createTextField()`, `MovieClip.getBounds()`, `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `MovieClip.getDepth()`, `MovieClip.getInstanceAtDepth()`, `MovieClip.getNextHighestDepth()`, `MovieClip.globalToLocal()`, `MovieClip.localToGlobal()`, `MovieClip.hitTest()`, `MovieClip.setMask()`, `MovieClip.swapDepths()`.

Per ulteriori informazioni su queste funzioni e questi metodi, vedere le relative voci nella *Guida di riferimento di ActionScript 2.0*.

Per un esempio di animazione con script in Flash è possibile trovare un file sorgente di esempio, `animation.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Animation*.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file immagine in un file SWF contenente animazione con script. È possibile trovare i file sorgente di esempio, `gallery_tree.fla` e `gallery_tween.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries*.

Chiamata di più metodi su un singolo clip filmato

È possibile utilizzare l'istruzione `with` per accedere a un clip filmato e quindi eseguire su di esso una serie di metodi. L'istruzione `with` è applicabile a tutti gli oggetti ActionScript (ad esempio Array, Color e Sound) e non solo ai clip filmato.

L'istruzione `with` accetta come parametro un clip filmato. L'oggetto specificato viene aggiunto alla fine del percorso target corrente. Tutte le azioni nidificate in un'istruzione `with` vengono eseguite all'interno del nuovo percorso target o area di validità. Ad esempio, nello script seguente, all'istruzione `with` viene passato l'oggetto `donut.hole` per modificare le proprietà di `hole`:

```
with (donut.hole) {  
    _alpha = 20;  
    _xscale = 150;  
    _yscale = 150;  
}
```

Lo script si comporta come se le istruzioni contenute nell'istruzione `with` fossero chiamate dalla linea temporale dell'istanza `hole`. Il codice riportato in precedenza è equivalente al seguente:

```
donut.hole._alpha = 20;  
donut.hole._xscale = 150;  
donut.hole._yscale = 150;
```

È equivalente anche al seguente:

```
with (donut) {  
    hole._alpha = 20;  
    hole._xscale = 150;  
    hole._yscale = 150;  
}
```

Caricamento e scaricamento di file SWF

Per riprodurre ulteriori file SWF senza chiudere Flash Player oppure per passare da un file SWF a un altro senza caricare una pagina HTML aggiuntiva, è possibile avvalersi di una delle seguenti possibilità:

- La funzione `loadMovie()` globale o il metodo `loadMovie()` della classe MovieClip.
- Il metodo `loadClip()` della classe MovieClipLoader. Per ulteriori informazioni sulla classe MovieClipLoader, vedere `MovieClipLoader` nella *Guida di riferimento di ActionScript 2.0*.

È inoltre possibile utilizzare il metodo `loadMovie()` per inviare variabili a uno script CGI che genera un file SWF come output CGI. Ad esempio, è possibile utilizzare questa procedura per caricare file SWF o di immagine dinamici in base a determinate variabili all'interno di un clip filmato. Quando si carica un file SWF, è possibile specificare come target un livello o un clip filmato in cui caricarlo. Se si carica un file SWF in un target, il file caricato eredita le proprietà del clip filmato identificato. Dopo aver caricato il filmato Flash, è possibile modificare queste proprietà.

Il metodo `unloadMovie()` rimuove un file SWF caricato in precedenza da `loadMovie()`. Lo scaricamento esplicito dei file SWF con il metodo `unloadMovie()` garantisce una transizione ininterrotta da un file all'altro e può ridurre la quantità di memoria richiesta da Flash Player. Può rivelarsi più efficiente in alcune situazioni impostare la proprietà `_visible` del clip su `false` anziché scaricare il clip. Se si prevedere di riutilizzare il clip in seguito, impostare la proprietà `_visible` su `false` e reimpostarla su `true` quando necessario.

Utilizzare `loadMovie()` per effettuare una delle seguenti operazioni:

- Riprodurre una sequenza di banner pubblicitari costituiti da file SWF inserendo una funzione `loadMovie()` in un file SWF contenitore che carica e scarica in modo sequenziale i file dei banner SWF.
- Sviluppare un'interfaccia a opzioni multiple con collegamenti che permettono all'utente di selezionare tra più file SWF che vengono utilizzati per visualizzare il contenuto di un sito.
- Creare un'interfaccia di navigazione con controlli nel livello 0 che consenta di caricare contenuto in altri livelli. Il caricamento di contenuto in livelli, rispetto al caricamento di nuove pagine HTML in un browser, permette di ottenere transizioni più fluide tra le pagine di contenuto.

Per ulteriori informazioni sul caricamento dei file SWF, vedere “[Caricamento di file SWF e file di immagine esterni](#)” a pagina 647.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Impostazione di una linea temporale principale per i file SWF caricati](#)” a pagina 386
- “[Caricamento dei file di immagine nei clip filmato](#)” a pagina 388

Impostazione di una linea temporale principale per i file SWF caricati

La proprietà ActionScript `_root` specifica o contiene un riferimento alla linea temporale principale di un file SWF. Se un file SWF ha più livelli, la linea temporale principale si trova sul livello contenente lo script in esecuzione. Ad esempio, se uno script nel livello 1 valuta `_root`, viene restituito `_level1`. Tuttavia, la linea temporale specificata da `_root` non è la stessa se un file SWF viene eseguito in modo indipendente (nel proprio livello) o se è stato caricato in un'istanza di clip filmato con una chiamata a `loadMovie()`.

Nell'esempio seguente, un file chiamato container.swf contiene un'istanza di clip filmato denominata `target_mc` nella linea temporale principale. Il file container.swf dichiara una variabile denominata `userName` nella propria linea temporale principale; lo stesso script carica quindi un altro file chiamato contents.swf nel clip filmato `target_mc`:

```
// In container.swf:  
_root.userName = "Tim";  
target_mc.loadMovie("contents.swf");  
my_btn.onRelease = function():Void {  
    trace(_root.userName);  
};
```

Nell'esempio seguente, anche il file SWF caricato, contents.swf, dichiara una variabile denominata `userName` nella linea temporale principale:

```
// In contents.swf:  
_root.userName = "Mary";
```

Dopo il caricamento del file contents.swf nel clip filmato in container.swf, il valore di `userName` associato alla linea temporale principale del file SWF contenitore (container.swf) viene impostato su “`Mary`” anziché “`Tim`”. In questo caso può verificarsi un malfunzionamento del codice in container.swf (e contents.swf).

Per impostare a `_root` la valutazione costante della linea temporale del file SWF caricato e non dell'effettiva linea temporale principale, utilizzare la proprietà `_lockroot`. Questa proprietà può essere impostata dal file SWF che esegue il caricamento oppure dal file SWF caricato. Quando la proprietà `_lockroot` è impostata su `true` in un'istanza di clip filmato, il relativo clip filmato agisce come `_root` per tutti i file SWF in esso caricati. Quando `_lockroot` è impostata su `true` in un file SWF, tale file agisce come proprio elemento principale (root), indipendentemente dal fatto che venga caricato da un altro file SWF. Qualsiasi clip filmato e qualsiasi numero di clip filmato può impostare `_lockroot` su `true`. Per impostazione predefinita, questa proprietà corrisponde a `false`.

L'autore di `container.swf`, ad esempio, può inserire il seguente codice nel fotogramma 1 della linea temporale principale:

```
// Aggiunto al fotogramma 1 in container.swf:  
target_mc._lockroot = true;
```

In questo modo si garantisce che tutti i riferimenti a `_root` in `contents.swf` o in qualsiasi file SWF caricato in `target_mc` si riferiscono alla loro linea temporale e non alla linea temporale principale effettiva di `container.swf`. Quindi, facendo clic sul pulsante, viene visualizzato "Tim".

In alternativa, l'autore di `contents.swf` può aggiungere il seguente codice alla linea temporale principale:

```
// Aggiunto al fotogramma 1 in contents.swf:  
this._lockroot = true;
```

In questo modo si garantisce che indipendentemente dalla posizione in cui viene caricato `contents.swf`, qualsiasi riferimento a `_root` punti alla propria linea temporale e non a quella del file SWF contenitore.

Per ulteriori informazioni, vedere `_lockroot` (`MovieClip._lockroot` property).

Caricamento dei file di immagine nei clip filmato

È possibile utilizzare la funzione `loadMovie()` o il metodo `MovieClip` con lo stesso nome per caricare file di immagine in un'istanza di clip filmato. È anche possibile usare la funzione `loadMovieNum()` per caricare un file di immagine in un livello.

Quando si carica un'immagine in un clip filmato, l'angolo superiore sinistro dell'immagine viene posizionato nel punto di registrazione del clip filmato. Poiché il punto di registrazione è spesso il centro del clip filmato, l'immagine caricata può non apparire centrata. Inoltre, quando si carica un'immagine sulla linea temporale principale, l'angolo superiore sinistro dell'immagine viene posizionato nell'angolo superiore sinistro dello stage. L'immagine caricata eredita la rotazione e la modifica in scala dal clip filmato, ma il contenuto originale del clip filmato viene rimosso.

Per ulteriori informazioni, consultare `loadMovie` function, `loadMovie` (`MovieClip.loadMovie` method) e `loadMovieNum` function nella *Guida di riferimento di ActionScript 2.0* e “[Caricamento di file SWF e file di immagine esterni](#)” a pagina [647](#).

Modifica della posizione e dell'aspetto di un clip filmato

Per modificare le proprietà di un clip filmato durante l'esecuzione, creare un'istruzione che assegna un valore a una proprietà o utilizzare la funzione `setProperty()`. Ad esempio, il codice seguente imposta la rotazione dell'istanza `mc` su 45:

```
my_mc._rotation = 45;
```

Il codice seguente consente di ottenere lo stesso risultato utilizzando la funzione `setProperty()`:

```
setProperty("my_mc", _rotation, 45);
```

I valori di alcune proprietà, dette *proprietà di sola lettura*, possono essere letti ma non impostati. Queste proprietà sono descritte come proprietà di sola lettura nelle relative voci della *Guida di riferimento di ActionScript 2.0*. Le seguenti sono proprietà di sola lettura: `_currentframe`, `_droptarget`, `_framesloaded`, `_parent`, `_target`, `_totalframes`, `_url`, `_xmouse` e `_ymouse`.

È possibile creare istruzioni per impostare le proprietà, a condizione che non siano di sola lettura. L'istruzione seguente imposta la proprietà `_alpha` dell'istanza di clip filmato `wheel_mc`, elemento secondario dell'istanza `car_mc`:

```
car_mc.wheel_mc._alpha = 50;
```

È inoltre possibile scrivere istruzioni che ottengano il valore di una proprietà di un clip filmato. Ad esempio, l'istruzione seguente ottiene il valore della proprietà `_xmouse` nella linea temporale del livello corrente e imposta la proprietà `_x` dell'istanza `my_mc` su tale valore:

```
this.onEnterFrame = function() {  
    my_mc._x = _root._xmouse;  
};
```

Il codice seguente consente di ottenere lo stesso risultato utilizzando la funzione `getProperty()`:

```
this.onEnterFrame = function() {  
    my_mc._x = getProperty(_root, '_xmouse');  
};
```

Le proprietà `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` e `_visible` sono influenzate dalle trasformazioni effettuate nell'elemento principale del clip filmato e modificano il clip filmato e tutti gli elementi secondari del clip. Le proprietà `_focusrect`, `_highquality`, `_quality` e `_soundbuftime` sono di tipo globale e appartengono solo alla linea temporale principale di livello 0. Tutte le altre proprietà sono relative a ogni clip filmato o livello caricato.

Per un elenco delle proprietà dei clip filmato, vedere il riepilogo delle proprietà per la classe MovieClip nella *Guida di riferimento di ActionScript 2.0*.

Per un esempio di animazione con script in Flash è possibile trovare un file sorgente di esempio, `animation.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript\Animation*.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file immagine in un file SWF contenente animazione con script. È possibile trovare i file sorgente di esempio, `gallery_tree.fla` e `gallery_tween.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript\Galleries*.

Trascinamento di clip filmato

È possibile utilizzare la funzione globale `startDrag()` o il metodo `MovieClip.startDrag()` per rendere trascinabile un clip filmato. Ad esempio, è possibile consentire il trascinamento di un clip filmato nel caso di giochi, funzioni di trascinamento della selezione, interfacce personalizzabili, barre di scorrimento e cursori.

Un clip rimane trascinabile fino a quando il trascinamento non viene interrotto esplicitamente mediante `stopDrag()` o fino a quando il trascinamento non viene associato a un altro clip filmato mediante `startDrag()`. È possibile trascinare in un file SWF un solo clip filmato alla volta.

Per creare funzionalità di trascinamento della selezione più complesse, è possibile valutare la proprietà `_droptarget` del clip filmato che viene trascinato. Si potrebbe, ad esempio, esaminare la proprietà `_droptarget` per verificare se il clip filmato è stato trascinato su un clip filmato specifico (ad esempio il clip filmato del cestino), quindi attivare un'altra azione, come nell'esempio seguente:

```
// Consente di trascinare un oggetto nel cestino.  
garbage_mc.onPress = function() {  
    this.startDrag(false);  
};  
// Quando l'oggetto si trova sul cestino, lo rende invisibile.  
garbage_mc.onRelease = function() {  
    this.stopDrag();  
    // Converte la notazione della barra in quella del punto tramite eval.  
    if (eval(this._droptarget) == trashcan_mc) {  
        garbage_mc._visible = false;  
    }  
};
```

Per ulteriori informazioni, vedere `startDrag` function o `startDrag` (`MovieClip.startDrag` method) nella *Guida di riferimento di ActionScript 2.0*.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare i clip filmato in modo dinamico mentre si caricano file immagine in un file SWF, operazione che comprende anche il trascinamento dei clip filmato. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `gallery_tween.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Galleries*.

Creazione di clip filmato in fase di runtime

Le istanze di clip filmato possono essere realizzate non solo nell'ambiente di creazione di Flash, ma anche in fase di runtime nei modi indicati di seguito:

- “[Creazione di un clip filmato vuoto](#)” a pagina 392
- “[Duplicazione o rimozione di un clip filmato](#)” a pagina 393
- “[Associazione di un simbolo clip filmato allo stage](#)” a pagina 394

Ogni istanza di clip filmato creata in fase di runtime deve avere un nome di istanza e un valore di profondità (impilamento o z-order). La profondità specificata determina in che modo il nuovo clip si sovrappone agli altri clip sulla stessa linea temporale. consente di sovrascrivere clip filmato presenti alla stessa profondità. Vedere “[Gestione delle profondità nei clip filmato](#)” a pagina 398.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare i clip filmato in modo dinamico mentre si caricano file immagine in un file SWF, operazione che comprende anche la creazione di clip filmato in fase di runtime. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, gallery_tween.fla.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries*.

Per un file sorgente di esempio che crea ed elimina diversi clip filmato in fase di runtime, nella cartella Samples del disco rigido è possibile trovare il file animation.fla.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation*.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Creazione di un clip filmato vuoto](#)” a pagina 392
- “[Duplicazione o rimozione di un clip filmato](#)” a pagina 393
- “[Associazione di un simbolo clip filmato allo stage](#)” a pagina 394

Creazione di un clip filmato vuoto

Per creare una nuova istanza di clip filmato vuota sullo stage, utilizzare il metodo `createEmptyMovieClip()` della classe `MovieClip`. Questo metodo crea un clip filmato secondario del clip che richiama il metodo. Il punto di registrazione di un clip filmato vuoto appena creato è l'angolo superiore sinistro.

Il seguente codice, ad esempio, crea un nuovo clip filmato secondario denominato `new_mc` a una profondità di 10 nel clip filmato denominato `parent_mc`.

```
parent_mc.createEmptyMovieClip("new_mc", 10);
```

Il seguente codice crea un nuovo clip filmato denominato `canvas_mc` nella linea temporale principale del file SWF in cui viene eseguito lo script e quindi chiama `loadMovie()` per caricare un file JPEG esterno.

```
this.createEmptyMovieClip("canvas_mc", 10);
canvas_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Come illustrato nell'esempio seguente, è possibile caricare l'immagine `image2.jpg` in un clip filmato e utilizzare il metodo `MovieClip.onPress()` per impostare l'immagine affinché si comporti come un pulsante. Se viene caricata un'immagine tramite `loadMovie()`, il clip filmato viene sostituito con l'immagine, ma non viene fornito accesso ai metodi del clip filmato. Per ottenere l'accesso ai metodi del clip filmato, creare un clip filmato principale vuoto e un clip filmato secondario contenitore. Caricare l'immagine nel contenitore e inserire il gestore di eventi nel clip filmato principale.

```
// Crea un clip filmato principale per ospitare il contenitore.
this.createEmptyMovieClip("my_mc", 0);
```

```
// Crea un clip filmato secondario all'interno di "my_mc".
// Il clip filmato che viene sostituito dall'immagine.
my_mc.createEmptyMovieClip("container_mc",99);
```

```
// Usa MovieClipLoader per caricare l'immagine.
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
    my_mc.container_mc);
```

```
// Inserisce il gestore di eventi nel clip filmato principale my_mc.
my_mc.onPress = function():Void {
    trace("It works");
};
```

Per ulteriori informazioni, vedere `createEmptyMovieClip` (`MovieClip.createEmptyMovieClip` method) nella *Guida di riferimento di ActionScript 2.0*.

Per un file sorgente di esempio che crea ed elimina diversi clip filmato in fase di runtime, nella cartella Samples del disco rigido è possibile trovare il file `animation.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation*.

Duplicazione o rimozione di un clip filmato

Per duplicare o rimuovere istanze di clip filmato, utilizzare le funzioni globali `duplicateMovieClip()` o `removeMovieClip()` oppure i metodi della classe `MovieClip` con lo stesso nome. Il metodo `duplicateMovieClip()` crea una nuova istanza da un'istanza di clip filmato esistente e le assegna un nuovo nome di istanza e una profondità, o *z-order*. I clip filmato duplicati iniziano sempre dal fotogramma 1, anche se il clip filmato originale si trovava in un fotogramma diverso al momento della duplicazione, e sono sempre in primo piano rispetto ai clip filmato definiti in precedenza nella linea temporale.

Per eliminare un clip filmato creato con `duplicateMovieClip()`, utilizzare `removeMovieClip()`. Anche i clip filmato duplicati vengono rimossi se viene eliminato il clip filmato principale.

Per ulteriori informazioni, vedere `duplicateMovieClip function` e `removeMovieClip function` nella *Guida di riferimento di ActionScript 2.0*.

Per un file sorgente di esempio che crea ed elimina diversi clip filmato in fase di runtime, nella cartella Samples del disco rigido è possibile trovare il file `animation.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation*.

Associazione di un simbolo clip filmato allo stage

Un ultimo modo per creare istanze di clip filmato in fase di runtime consiste nell'utilizzare il metodo `attachMovie()`. Il metodo `attachMovie()` associa allo stage un'istanza del simbolo clip filmato nella libreria del file SWF. Il nuovo clip diventa un clip secondario del clip che lo ha associato.

Per utilizzare ActionScript per associare un simbolo clip filmato dalla libreria, esportare il simbolo per ActionScript e assegnarvi un identificatore di concatenamento univoco. Per eseguire questa operazione, utilizzare la finestra di dialogo Proprietà di concatenamento.

Per impostazione predefinita, tutti i clip filmato esportati per l'utilizzo in ActionScript vengono caricati prima del fotogramma iniziale del file SWF che li contiene. Questo potrebbe comportare un ritardo nell'esecuzione del primo fotogramma. Quando si assegna un identificatore di concatenamento a un elemento, è possibile specificare se il contenuto deve essere aggiunto prima del primo fotogramma. In caso contrario, è necessario includerne un'istanza in altri fotogrammi del file SWF, altrimenti l'elemento non viene esportato nel file SWF.

Per assegnare un identificatore di concatenamento a un clip filmato:

1. Selezionare Finestra > Libreria per aprire il pannello Libreria.
2. Selezionare un clip filmato nel pannello Libreria.
3. Nel pannello Libreria, scegliere Concatenamento dal menu a comparsa.
Viene visualizzata la finestra di dialogo Proprietà del concatenamento.
4. In Concatenamento, selezionare Esporta per ActionScript.
5. In Identificatore, immettere un ID per il clip filmato.

Per impostazione predefinita, l'identificatore è uguale al nome del simbolo.

È possibile, se lo si desidera, assegnare una classe ActionScript al simbolo del clip filmato affinché il clip filmato erediti i metodi e le proprietà di una determinata classe. Vedere “[Assegnazione di una classe a un simbolo clip filmato](#)” a pagina 412.

6. Se non si desidera che il clip filmato venga caricato prima del fotogramma iniziale, deselectrare l'opzione Esporta nel primo fotogramma.
In questo caso, inserire un'istanza del clip filmato sul fotogramma della linea temporale in cui si desidera rendere disponibile il clip filmato. Se, ad esempio, lo script che si crea non fa riferimento al clip filmato fino al fotogramma 10, inserire un'istanza del simbolo sulla linea temporale in una posizione precedente o corrispondente al fotogramma 10.
7. Fare clic su OK.

Dopo avere assegnato un identificatore di concatenamento a un clip filmato, è possibile associare un'istanza del simbolo allo stage in fase di runtime utilizzando `attachMovie()`.

Per assegnare un clip filmato a un altro clip filmato:

1. Assegnare un identificatore di concatenamento a un simbolo della libreria del clip filmato, come illustrato nell'esempio precedente.
2. Mantenendo aperto il pannello Azioni (Finestra > Azioni), selezionare un fotogramma nella linea temporale.
3. Nel riquadro Script del pannello Azioni, digitare il nome del clip filmato o del livello a cui si desidera associare il nuovo clip filmato.
Ad esempio, per associare il clip filmato alla linea temporale principale, digitare **this**.

4. Nella casella degli strumenti Azioni (a sinistra del pannello Azioni), selezionare Classi ActionScript 2.0 > Filmato > MovieClip > Metodi e scegliere **attachMovie()**.

5. Utilizzando i suggerimenti sul codice visualizzati come guida, immettere i valori per i seguenti parametri:
 - Per **IDname**, specificare il nome dell'identificatore immesso nella finestra di dialogo Proprietà del concatenamento.
 - Per **newName**, immettere un nome di istanza per il clip associato, in modo che sia possibile farvi riferimento.
 - Per **depth**, immettere il livello nel quale il clip filmato duplicato verrà associato al clip filmato. Ogni clip filmato associato ha un proprio ordine di impilamento, con il livello 0 come livello del filmato di origine. I clip filmato associati sono sempre in primo piano rispetto al clip filmato originale, come illustrato nell'esempio seguente:
`this.attachMovie("calif_id", "california_mc", 10);`

Per ulteriori informazioni, vedere **attachMovie (MovieClip.attachMovie method)** nella *Guida di riferimento di ActionScript 2.0*.

Aggiunta di parametri a clip filmato creati dinamicamente

Se si crea o si duplica dinamicamente un clip filmato utilizzando `MovieClip.attachMovie()` e `MovieClip.duplicateMovie()`, è possibile completare il clip filmato con i parametri di un altro oggetto. Il parametro `initObject` di `attachMovie()` e `duplicateMovie()` consente ai clip filmato creati in modo dinamico di ricevere i parametri del clip.

Per ulteriori informazioni, vedere `attachMovie` (`MovieClip.attachMovie` method) e `duplicateMovieClip` (`MovieClip.duplicateMovieClip` method) nella *Guida di riferimento di ActionScript 2.0*.

Per completare un clip filmato creato in modo dinamico con i parametri di un oggetto specificato:

Effettuare una delle seguenti operazioni:

- Utilizzare la seguente sintassi con `attachMovie()`:

```
myMovieClip.attachMovie(idName, newName, depth [, initObject]);
```

- Adottare la seguente sintassi con `duplicateMovie()`:

```
myMovieClip.duplicateMovie(idName, newName, depth [, initObject]);
```

Il parametro `initObject` specifica il nome dell'oggetto di cui si desidera utilizzare i parametri per completare il clip filmato creato in modo dinamico.

Per completare un clip filmato con i parametri usando il metodo `attachMovie()`:

1. In un nuovo documento Flash, creare un simbolo clip filmato selezionando Inserisci > Nuovo simbolo.
2. Digitare **dynamic_mc** nella casella di testo Nome simbolo e selezionare il comportamento Clip filmato.
3. Nel simbolo, creare un campo di testo dinamico sullo stage con il nome di istanza **name_txt**.
Questo campo di testo deve essere inserito in basso e a destra rispetto al punto di registrazione.
4. Selezionare il primo fotogramma della linea temporale del clip filmato e aprire il pannello Azioni (Finestra > Azioni).
5. Creare una nuova variabile denominata `name_str`, quindi assegnarne il valore alla proprietà `text` di `name_txt`, come nell'esempio seguente:

```
var name_str:String;
name_txt.text = name_str;
```

6. Selezionare Modifica > Modifica documento per tornare alla linea temporale principale.
7. Selezionare il simbolo del clip filmato nella libreria, quindi Concatenamento dal menu a comparsa del pannello Libreria.
Viene visualizzata la finestra di dialogo Proprietà del concatenamento.
8. Selezionare Esporta per ActionScript ed Esporta nel primo fotogramma.
9. Digitare **dynamic_id** nella casella di testo Identificatore e fare clic su OK.
10. Selezionare il primo fotogramma della linea temporale principale e aggiungere il codice seguente al riquadro Script del pannello Azioni:

```
/* Associa un nuovo clip filmato e lo sposta alle coordinate x e y 50 */
this.attachMovie("dynamic_id", "newClip_mc", 99, {name_str:"Erick",
_x:50, _y:50});
```

11. Provare il documento Flash selezionando Controllo > Prova filmato.

Il nome specificato nella chiamata `attachMovie()` è visualizzato all'interno del campo di testo del nuovo clip filmato.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare i clip filmato in modo dinamico mentre si caricano file immagine in un file SWF, operazione che comprende anche la creazione di clip filmato in fase di runtime. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `gallery_tween.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries*.

Per un file sorgente di esempio che crea ed elimina diversi clip filmato in fase di runtime, nella cartella Samples del disco rigido è possibile trovare il file `animation.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation*.

Gestione delle profondità nei clip filmato

Ogni clip filmato ha un proprio spazio *z-order* che determina in che modo gli oggetti si sovrappongono all'interno del file SWF o del clip filmato di origine. A ciascun clip filmato è associato un valore di profondità che determina se il filmato si trova in primo piano o dietro altri clip filmato della stessa linea temporale. Quando si crea un clip filmato in fase di runtime attraverso `attachMovie` (`MovieClip.attachMovie` method), `duplicateMovieClip` (`MovieClip.duplicateMovieClip` method) o `createEmptyMovieClip` (`MovieClip.createEmptyMovieClip` method), occorre sempre specificare una profondità per il nuovo clip come parametro del metodo. Ad esempio, il codice seguente associa un nuovo clip filmato alla linea temporale di un clip filmato denominato `container_mc` con valore di profondità pari a 10.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
```

Questo esempio crea un nuovo clip filmato con una profondità di 10 all'interno dello spazio *z-order* di `container_mc`.

Il codice seguente associa due nuovi clip filmato a `container_mc`. Il rendering del primo clip, denominato `clip1_mc`, avviene dietro `clip2_mc`, perché il valore di profondità assegnato a questo clip è inferiore.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
container_mc.attachMovie("symbolID", "clip2_mc", 15);
```

I valori di profondità per i clip filmato possono variare da -16384 a 1048575. Se si crea o si associa un nuovo clip filmato a una profondità in cui è già presente un altro clip filmato, il nuovo clip o il clip associato sovrascrive il contenuto esistente. Per evitare l'insorgere di questo problema, utilizzare il metodo `MovieClip.getNextHighestDepth()`; tuttavia, non si deve sfruttare questo metodo con componenti che impiegano un sistema di gestione della profondità diverso. Con istanze di componente, utilizzare invece “Classe DepthManager”.

La classe `MovieClip` fornisce diversi metodi per la gestione delle profondità dei clip filmato; per ulteriori informazioni consultare `getNextHighestDepth` (`MovieClip.getNextHighestDepth` method), `getInstanceAtDepth` (`MovieClip.getInstanceAtDepth` method), `getDepth` (`MovieClip.getDepth` method) e `swapDepths` (`MovieClip.swapDepths` method) nella *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni sulle profondità dei clip filmato, consultare i seguenti argomenti:

- “[Determinazione della successiva profondità massima disponibile](#)” a pagina 399
- “[Determinazione dell'istanza a una profondità particolare](#)” a pagina 399
- “[Determinazione della profondità di un'istanza](#)” a pagina 400
- “[Scambio delle profondità dei clip filmato](#)” a pagina 400

Determinazione della successiva profondità massima disponibile

Per determinare la successiva profondità massima disponibile in un clip filmato, utilizzare `MovieClip.getNextHighestDepth()`. Il valore intero restituito da questo metodo indica la successiva profondità disponibile che determina la posizione in primo piano rispetto a tutti gli altri oggetti del clip filmato.

Il codice seguente associa un nuovo clip filmato, con un valore di profondità pari a 10, sulla linea temporale principale, `file_mc`, quindi determina la profondità massima successiva disponibile nello stesso clip filmato e crea un nuovo clip filmato denominato `edit_mc` a tale profondità.

```
this.attachMovie("menuClip","file_mc", 10, {_x:0, _y:0});  
trace(file_mc.getDepth()); // 10  
var nextDepth:Number = this.getNextHighestDepth();  
this.attachMovie("menuClip", "edit_mc", nextDepth, {_x:200, _y:0});  
trace(edit_mc.getDepth()); // 11
```

In questo caso, la variabile denominata `nextDepth` contiene il valore 11, in quanto si tratta della successiva profondità massima disponibile per il clip filmato `edit_mc`.

Non utilizzare `MovieClip.getNextHighestDepth()` con i componenti; per contro, sfruttare il gestore di profondità. Per ulteriori informazioni, vedere “Classe DepthManager” nella *Guida di riferimento dei componenti*. Per ulteriori informazioni su `MovieClip.getNextHighestDepth()`, vedere `getNextHighestDepth` (`MovieClip.getNextHighestDepth` method).

Per ottenere la profondità massima occupata, sottrarre 1 dal valore restituito da `getNextHighestDepth()`, come descritto nella sezione successiva.

Determinazione dell'istanza a una profondità particolare

Per determinare l'istanza presente a una particolare profondità, utilizzare `MovieClip.getInstanceAtDepth()`. Questo metodo restituisce un riferimento all'istanza `MovieClip` nella profondità specificata.

Il seguente codice combina `getNextHighestDepth()` e `getInstanceAtDepth()` per determinare il clip filmato nella profondità massima corrente occupata della linea temporale principale.

```
var highestOccupiedDepth:Number = this.getNextHighestDepth() - 1;  
var instanceAtHighestDepth:MovieClip =  
    this.getInstanceAtDepth(highestOccupiedDepth);
```

Per ulteriori informazioni, vedere `getInstanceAtDepth` (`MovieClip.getInstanceAtDepth` method) nella *Guida di riferimento di ActionScript 2.0*.

Determinazione della profondità di un'istanza

Per determinare la profondità di un'istanza di clip filmato, utilizzare `MovieClip.getDepth()`.

Il seguente codice esegue le iterazioni sui clip filmato nella linea temporale principale del file SWF e visualizza il nome di istanza e il valore di profondità di ogni filmato nel pannello Output.

```
for (var item:String in _root) {  
    var obj:Object = _root[item];  
    if (obj instanceof MovieClip) {  
        var objDepth:Number = obj.getDepth();  
        trace(obj._name + ":" + objDepth)  
    }  
}
```

Per ulteriori informazioni, vedere `getDepth` (`MovieClip.getDepth` method) nella *Guida di riferimento di ActionScript 2.0*.

Scambio delle profondità dei clip filmato

Per scambiare le profondità di due clip filmato sulla stessa linea temporale, utilizzare `MovieClip.swapDepths()`. I prossimi esempi mostrano come due istanze di clip filmato si scambiano le profondità in fase di runtime.

Per scambiare le profondità nei clip filmato:

1. Creare un nuovo documento Flash denominato **swap.fla**.
2. Disegnare un cerchio blu sullo stage.
3. Selezionare il cerchio blu e scegliere Elabora > Converti in simbolo.
4. Selezionare l'opzione Clip filmato e fare clic su OK.
5. Selezionare l'istanza nello stage e digitare **first_mc** nella casella di testo Nome istanza della finestra di ispezione Proprietà.
6. Disegnare un cerchio rosso sullo stage e selezionare Elabora > Converti in simbolo.
7. Selezionare l'opzione Clip filmato e fare clic su OK.
8. Selezionare l'istanza nello stage e digitare **second_mc** nella casella di testo Nome istanza della finestra di ispezione Proprietà.
9. Trascinare le due istanze così che si sovrappongano leggermente sullo stage.

- 10.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
first_mc.onRelease = function() {
    this.swapDepths(second_mc);
};

second_mc.onRelease = function() {
    this.swapDepths(first_mc);
};
```

- 11.** Selezionare Controllo > Prova filmato per provare il documento.

Quando si fa clic sulle istanze presenti nello stage, queste si scambiano le profondità. Si vedranno le due istanze cambiare il clip che si trova sopra l'altro clip.

Per ulteriori informazioni, vedere `swapDepths` (`MovieClip.swapDepths` method) nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sulla memorizzazione nella cache e sullo scorrimento dei clip filmato in ActionScript

Man mano che i progetti di Flash si ingrandiscono, indipendentemente dal fatto che si stia creando un'applicazione o animazioni con script complesse, è necessario prendere in considerazione le prestazioni e l'ottimizzazione. Quando si dispone di contenuto che rimane statico (ad esempio un clip filmato rettangolo), Flash non ottimizza il contenuto. Pertanto, quando si cambia la posizione del clip filmato rettangolo, Flash deve ridisegnare tutto il rettangolo (in Flash Player 7 e versioni precedenti).

In Flash Player 8, è possibile memorizzare clip filmato e pulsanti specifici nella cache in modo da migliorare le prestazioni del file SWF. Il clip filmato o il pulsante sono una *superficie*, essenzialmente una versione bitmap dei dati vettoriali dell'istanza, vale a dire dei dati che non si intende modificare in modo significativo nel corso del file SWF. Pertanto, le istanze per cui è stata attivata la memorizzazione nella cache non vengono continuamente ridisegnate durante le riproduzione del file SWF, il che consente un rendering più rapido del file.



I dati vettoriali possono essere aggiornati e quando si esegue l'aggiornamento la superficie viene ricreata. Per questo motivo, i dati vettoriali memorizzati nella cache nella superficie non devono restare immutati per tutto il file SWF.

È possibile usare ActionScript per abilitare la memorizzazione nella cache, lo scorrimento e il controllo degli sfondi. Per attivare la memorizzazione nella cache di un'istanza di clip filmato, è possibile usare la finestra di ispezione Proprietà. Per memorizzare nella cache clip filmato o pulsanti senza utilizzare ActionScript, è possibile selezionare l'opzione Usa caching bitmap in runtime nella finestra di ispezione Proprietà.

La tabella seguente contiene brevi descrizioni delle nuove proprietà per le istanze di clip filmato:

Proprietà	Descrizione
cacheAsBitmap	L'istanza del clip filmato memorizza nella cache una rappresentazione bitmap di se stessa. Flash crea un oggetto di superficie per l'istanza, rappresentato da bitmap memorizzate nella cache invece che da dati vettoriali. Se si cambiano i limiti del clip filmato, la superficie viene ricreata anziché ridimensionata. Per ulteriori informazioni e un esempio, vedere "Memorizzazione di un clip filmato nella cache" a pagina 406 .
opaqueBackground	Consente di specificare un colore di sfondo per l'istanza di clip filmato opaque. Se questa proprietà viene impostata su un valore numerico, l'istanza del clip filmato presenta una superficie opaca (non trasparente). Una bitmap opaca non dispone di un canale alfa (trasparenza) e il suo rendering viene eseguito più rapidamente. Per ulteriori informazioni e un esempio, vedere "Impostazione dello sfondo di un clip filmato" a pagina 408 .
scrollRect	Questa proprietà consente di scorrere velocemente il contenuto del clip filmato per ottenere una visualizzazione più estesa del contenuto nella finestra. Al contenuto del clip filmato viene applicato il ritaglio, e lo scorrimento dell'istanza avviene con offset di scorrimento, altezza e larghezza specifici. Questa proprietà consente di scorrere velocemente il contenuto del clip filmato per ottenere una finestra che visualizza più contenuto rispetto all'area dello stage. I campi di testo e il contenuto complesso che viene visualizzato nell'istanza possono scorrere più velocemente in quanto Flash non deve ricreare i dati vettoriali di tutto il clip filmato. Per ulteriori informazioni e un esempio, vedere scrollRect (MovieClip.scrollRect property).

Queste tre proprietà sono indipendenti una dall'altra; tuttavia, le proprietà opaqueBackground e scrollRect funzionano meglio quando un oggetto viene memorizzato nella cache come bitmap. I miglioramenti nelle prestazioni si noteranno solo per le proprietà opaqueBackground e scrollRect quando si imposta cacheAsBitmap su true.

Per creare una superficie che sia anche possibile scorrere, occorre impostare le proprietà `cacheAsBitmap` e `scrollRect` per l'istanza del clip filmato. Le superfici possono essere nidificate all'interno di altre. La superficie copia la bitmap nella superficie principale.

Per informazioni sull'effetto maschera del canale alfa, che richiede l'impostazione della proprietà `cacheAsBitmap` su `true`, vedere [“Applicazione dell'effetto maschera ai canali alfa” a pagina 411](#).

NOTA

Non è possibile applicare la memorizzazione nella cache direttamente ai campi di testo. Per sfruttare al meglio questa funzione, è necessario disporre il testo all'interno di un clip filmato. Per un esempio, consultare il file di esempio nella *directory di installazione di Flash\Samples and Tutorials\Samples\ActionScript\FlashType*.

È possibile trovare un file sorgente di esempio in cui viene descritto come applicare a un'istanza la memorizzazione delle bitmap nella cache. Individuare il file denominato `cacheBitmap.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

È anche possibile trovare un file sorgente di esempio in cui viene descritto come applicare al testo scorrevole la memorizzazione delle bitmap nella cache. Individuare il file denominato `flashtype.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

Quando attivare la memorizzazione nella cache

L'attivazione della funzione di memorizzazione nella cache di un clip filmato consente di creare una superficie, il che presenta diversi vantaggi, fra cui quello di consentire il rendering più rapido delle animazioni vettoriali complesse. L'attivazione della memorizzazione nella cache può essere utile in diversi casi. Nonostante questa funzione possa sembrare utile per migliorare le prestazioni dei file SWF in tutti i casi, ci sono situazioni in cui questa funzione non solo non migliora le prestazioni, ma, anzi, le può peggiorare. In questa sezione sono descritti i vari scenari in cui utilizzare la memorizzazione nella cache e i casi in cui è opportuno invece usare i normali clip filmato.

Le prestazioni generali dei dati memorizzati nella cache dipendono dalla complessità dei dati vettoriali delle istanze, dalla quantità di dati modificati e dal fatto che sia stata impostata o meno la proprietà `opaqueBackground`. Se si stanno modificando delle aree piccole, la differenza tra l'uso di una superficie e l'uso dei dati vettoriali è minima. Prima di distribuire l'applicazione, potrebbe essere utile provare entrambi gli scenari con il proprio lavoro.

Per informazioni sull'effetto maschera del canale alfa, che richiede l'impostazione della proprietà `cacheAsBitmap` su `true`, vedere [“Applicazione dell'effetto maschera ai canali alfa” a pagina 411](#).

Quando utilizzare la memorizzazione delle bitmap nella cache

Segue una descrizione degli scenari in cui l'attivazione della memorizzazione delle bitmap nella cache può comportare notevoli vantaggi.

Immagine di sfondo complessa Un'applicazione contenente un'immagine di sfondo complessa e dettagliata di dati vettoriali (magari un'immagine in cui è stato applicato il comando Ricalca bitmap o un'immagine creata in Adobe Illustrator). È possibile animare i personaggi sullo sfondo, operazione che però rallenta l'animazione perché lo sfondo deve continuamente rigenerare i dati vettoriali. Per migliorare le prestazioni è possibile selezionare il contenuto, memorizzarlo in un clip filmato e impostare la proprietà `opaqueBackground` su `true`. Viene eseguito il rendering dello sfondo come bitmap così che possa essere ridisegnato velocemente; in questo modo l'animazione viene riprodotta più rapidamente.

Campo di testo a scorrimento Un'applicazione che visualizza una grande quantità di testo in un campo di testo a scorrimento. È possibile inserire il campo di testo in un clip filmato impostato come scorrevole con contorni a scorrimento (la proprietà `scrollRect`). In questo modo si attiva lo scorrimento veloce dei pixel per l'istanza specificata. Quando un utente esegue lo scorrimento dell'istanza di clip filmato, Flash sposta verso l'alto i pixel già visualizzati e genera l'area che viene mostrata invece di rigenerare l'intero campo di testo.

Sistema a finestre Un'applicazione con un complesso sistema di finestre che si sovrappongono. Ciascuna finestra può essere aperta o chiusa (per esempio, le finestre dei browser Web). Se si contrassegna ciascuna finestra come superficie (impostare la proprietà `cacheAsBitmap` su `true`), ogni finestra viene isolata e memorizzata nella cache. Gli utenti possono trascinare le finestre così che si sovrappongano tra loro, e ciascuna finestra non deve rigenerare il contenuto vettoriale.

Tutti questi scenari migliorano la sensibilità e l'interattività dell'applicazione ottimizzando le immagini vettoriali.

È possibile trovare un file sorgente di esempio in cui viene descritto come applicare a un'istanza la memorizzazione delle bitmap nella cache. Individuare il file denominato cacheBitmap.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

È anche possibile trovare un file sorgente di esempio in cui viene descritto come applicare al testo scorrevole la memorizzazione delle bitmap nella cache. Individuare il file denominato flashtype.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

Quando evitare di utilizzare la memorizzazione delle bitmap nella cache

L'uso scorretto di questa funzione può incidere in modo negativo sul file SWF. Quando si sviluppa un file FLA che usa le superfici, ricordare queste linee guida:

- Non usare in modo eccessivo le superfici (clip filmato con memorizzazione nella cache abilitata). Ogni superficie usa più memoria di un clip filmato normale; pertanto, si consiglia di abilitare le superfici solo quando è necessario migliorare le prestazioni del rendering.
Una bitmap memorizzata nella cache può utilizzare molta più memoria di una normale istanza di clip filmato. Per esempio, se il clip filmato sullo stage misura 250 pixel per 250 pixel, quando memorizzato nella cache utilizza 250 KB invece di 1 KB, situazione che si verifica se è un'istanza di clip filmato normale (non memorizzata nella cache).
- Evitare di ingrandire le superfici memorizzate nella cache. Se si abusa della funzione di memorizzazione delle bitmap nella cache, si consuma una grande quantità di memoria (vedere il punto precedente) soprattutto se si aumenta il contenuto.
- Usare le superfici per le istanze di clip filmato che sono prevalentemente statiche (prive di animazione). È possibile trascinare o spostare l'istanza, ma il contenuto della stessa non dovrebbe animarsi né risultare molto modificato. Per esempio, se si ruota o si trasforma un'istanza, questa passa da superficie a dati vettoriali, cosa che rende difficile l'elaborazione e influisce negativamente sul file SWF.

- Se si mescolano le superfici con i dati vettoriali, si aumenta la quantità di elaborazione che deve eseguire Flash Player (e a volte il computer). Si consiglia di raggruppare il più possibile insieme le superfici; per esempio, quando si creano applicazioni a finestra.

Memorizzazione di un clip filmato nella cache

Per memorizzare un'istanza di clip filmato nella cache, occorre impostare la proprietà `cacheAsBitmap` su `true`. Può accadere che, dopo aver impostato la proprietà `cacheAsBitmap` su `true`, l'istanza del clip filmato esegua automaticamente l'aggancio ai pixel su tutte le coordinate. Quando si prova il file SWF, si può notare che il rendering delle animazioni vettoriali complesse viene eseguito più velocemente.

Se si verifica una delle condizioni elencate di seguito, la superficie (bitmap memorizzata nella cache) non viene creata anche se `cacheAsBitmap` è impostata su `true`:

- La larghezza o l'altezza della bitmap è superiore a 2880 pixel.
- La bitmap non viene allocata (errore di memoria esaurita).

Per memorizzare un clip filmato nella cache:

1. Creare un nuovo documento Flash e denominarlo `cachebitmap.fla`.
2. Digitare 24 nella casella di testo `fps` della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
3. Creare o importare nel file FLA una grafica vettoriale complessa.
Nella seguente directory è possibile trovare un'immagine vettoriale complessa nel file sorgente completo per questo esempio:
 - In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap*.
 - In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/CacheBitmap*.
4. Selezionare l'immagine vettoriale e scegliere Elabora > Converti in simbolo.
5. Digitare `star` nella casella di testo Nome e poi fare clic su Avanzato (se la finestra di dialogo non è già ingrandita).
6. Selezionare Esporta per ActionScript (che seleziona anche Esporta nel primo fotogramma).
7. Digitare `star_id` nella casella di testo Identificatore.
8. Fare clic su OK per creare il simbolo del clip filmato, con l'indicatore di concatenamento Star.

- 9.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```
import mx.transitions.Tween;

var star_array:Array = new Array();
for (var i:Number = 0; i < 20; i++) {
    makeStar();
}
function makeStar():Void {
    var depth:Number = this.getNextHighestDepth();
    var star_mc:MovieClip = this.attachMovie("star_id", "star" + depth,
    depth);
    star_mc.onEnterFrame = function() {
        star_mc._rotation += 5;
    }
    star_mc._y = Math.round(Math.random() * Stage.height - star_mc._height
    / 2);
    var star_tween:Tween = new Tween(star_mc, "_x", null, 0, Stage.width,
    (Math.random() * 5) + 5, true);
    star_tween.onMotionFinished = function():Void {
        star_tween.yoyo();
    };
    star_array.push(star_mc);
}
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
    var star_mc:MovieClip;
    for (var i:Number = 0; i < star_array.length; i++) {
        star_mc = star_array[i];
        star_mc.cacheAsBitmap = !star_mc.cacheAsBitmap;
    }
}
Mouse.addListener(mouseListener);
```

- 10.** Selezionare Controllo > Prova filmato per provare il documento.

- 11.** Fare clic in qualsiasi punto dello stage per attivare la memorizzazione della bitmap nella cache.

Si noterà che l'animazione, invece di cambiare 1 fotogramma ogni secondo, appare più fluida poiché le istanze si muovono avanti e indietro sullo stage. Quando si fa clic sullo stage, l'impostazione di `cacheAsBitmap` passa da `true` a `false` e viceversa.

Se si attiva e disattiva la memorizzazione nella cache, come illustrato nell'esempio precedente, i dati presenti nella cache vengono liberati. È anche possibile applicare questo codice a un'istanza Button. Vedere `cacheAsBitmap` (`Button.cacheAsBitmap` property) nella *Guida di riferimento di ActionScript 2.0*.

Per esempi di scorimento dei clip filmato, vedere `scrollRect` (`MovieClip.scrollRect` property) nella *Guida di riferimento di ActionScript 2.0*. Per informazioni sull'effetto maschera del canale alfa, che richiede l'impostazione della proprietà `cacheAsBitmap` su `true`, vedere “[Applicazione dell'effetto maschera ai canali alfa](#)” a pagina 411.

È possibile trovare un file sorgente di esempio in cui viene descritto come applicare a un'istanza la memorizzazione delle bitmap nella cache. Individuare il file denominato `cacheBitmap.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

È anche possibile trovare un file sorgente di esempio in cui viene descritto come applicare al testo scorrevole la memorizzazione delle bitmap nella cache. Individuare il file denominato `flashtype.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

Impostazione dello sfondo di un clip filmato

È possibile impostare uno sfondo opaco per un clip filmato. Per esempio, quando si dispone di uno sfondo contenente un'immagine vettoriale complessa, è possibile impostare la proprietà `opaqueBackground` su un colore specifico (solitamente lo stesso colore dello stage). Lo sfondo viene quindi trattato come una bitmap, che consente di ottimizzare le prestazioni.

Quando si impone `cacheAsBitmap` su `true` e `opaqueBackground` su un colore specifico, la proprietà `opaqueBackground` consente alla bitmap interna di essere opaca e di velocizzare l'operazione di rendering. Se `cacheAsBitmap` non viene impostata su `true`, la proprietà `opaqueBackground` aggiunge una forma vettoriale quadrata opaca allo sfondo dell'istanza del clip filmato. Non crea automaticamente una bitmap.

L'esempio seguente mostra come impostare lo sfondo di un clip filmato per ottimizzare le prestazioni.

Per impostare lo sfondo di un clip filmato:

1. Creare un nuovo documento Flash denominato **background.fla**.
2. Disegnare un cerchio blu sullo stage.
3. Selezionare il cerchio blu e scegliere Elabora > Converti in simbolo.
4. Selezionare l'opzione Clip filmato e fare clic su OK.
5. Selezionare l'istanza nello stage e digitare **my_mc** nella casella di testo Nome istanza della finestra di ispezione Proprietà.
6. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
/* When you set cacheAsBitmap, the internal bitmap is opaque and renders
   faster. */
my_mc.cacheAsBitmap = true;
my_mc.opaqueBackground = 0xFF0000;
```

7. Selezionare Controllo > Prova filmato per provare il documento.

Il clip filmato appare sullo stage con il colore di sfondo specificato.

Per ulteriori informazioni su questa proprietà, vedere [opaqueBackground](#) (`MovieClip.opaqueBackground` property) nella *Guida di riferimento di ActionScript 2.0*.

Uso di clip filmato come maschere

È possibile utilizzare un clip filmato come maschera per creare un'area trasparente attraverso cui sia visibile il contenuto di un altro clip. Il clip filmato maschera riproduce tutti i fotogrammi nella relativa linea temporale come un clip filmato standard. È possibile rendere trascinabile un clip filmato usato come maschera, animarlo lungo una guida di movimento, utilizzare forme separate all'interno di una singola maschera o ridimensionare dinamicamente una maschera. È inoltre possibile utilizzare codice ActionScript per attivare e disattivare una maschera.

Non è possibile utilizzare una maschera per mascherarne un'altra, né impostare la proprietà `_alpha` di un clip filmato utilizzato come maschera. In un clip filmato impostato come maschera vengono utilizzati solo i riempimenti, mentre i tratti vengono ignorati.

Per creare una maschera:

1. Creare un quadrato sullo stage mediante lo strumento Rettangolo.
2. Selezionare il quadrato e premere F8 per convertirlo in un clip filmato.
L'istanza rappresenta la maschera.
3. Nella finestra di ispezione Proprietà, digitare **mask_mc** nella casella di testo Nome istanza.

Il clip filmato mascherato verrà rivelato sotto tutte le aree opache (non trasparenti) del clip filmato che agisce da maschera.

4. Selezionare il fotogramma 1 nella linea temporale.
5. Aprire il pannello Azioni (Finestra > Azioni) se non già visualizzato.
6. Nel pannello Azioni, inserire il codice seguente:

```
System.security.allowDomain("http://www.helpexamples.com");

this.createEmptyMovieClip("img_mc", 10);
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip):Void {
    target_mc.setMask(mask_mc);
}
var my_mc:MovieClipLoader = new MovieClipLoader();
my_mc.addListener(mcListener);
my_mc.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);
```

7. Selezionare Controllo > Prova filmato per provare il documento.

Un'immagine JPEG esterna viene caricata nel file SWF in fase di runtime e verrà sottoposta a mascheratura mediante la forma disegnata in precedenza sullo stage.

Per ulteriori informazioni, vedere `setMask (MovieClip.setMask method)` nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sulla mascheratura di caratteri dispositivo

È possibile utilizzare un clip filmato per mascherare il testo impostato con un carattere dispositivo. Un clip filmato maschera su un carattere dispositivo può funzionare correttamente solo se l'utente dispone di Flash Player 6 (6.0.40.0) o una versione successiva.

Se si utilizza un clip filmato per mascherare un testo impostato in un carattere dispositivo, il riquadro di delimitazione rettangolare della maschera viene utilizzato come area di mascheratura. Questo significa che se per un testo con un carattere dispositivo si crea una maschera di clip filmato di forma non rettangolare nell'ambiente di creazione Flash, la maschera che appare nel file SWF assume la forma del riquadro rettangolare di delimitazione e non quella della maschera stessa.

È possibile mascherare i caratteri dispositivo solo utilizzando un clip filmato come maschera, e non invece aggiungendo un livello maschera sullo stage.

Applicazione dell'effetto maschera ai canali alfa

L'applicazione dell'effetto maschera ai canali alfa viene supportato se sia i clip filmato di maschera che quelli mascherati utilizzano la memorizzazione delle bitmap nella cache. Questo supporto consente anche di utilizzare un filtro sulla maschera indipendentemente dal filtro applicato alla maschera stessa.

Per vedere un esempio dell'applicazione dell'effetto maschera ai canali alfa, scaricare il file di esempio all'indirizzo www.macromedia.com/go/flash_samples.

In questo file di esempio, la maschera è un ovale (`oval_mask`) con un valore alfa del 50% e un filtro di sfocatura. L'elemento mascherato (`flower_maskee`) ha un valore alfa del 100% e non gli è stato applicato nessun filtro. A entrambi i clip filmato è stata applicata la memorizzazione delle bitmap nella cache in fase di runtime nella finestra di ispezione Proprietà.

Nel pannello Azioni, il codice seguente è posizionato nel fotogramma 1 della linea temporale:

```
flower_maskee.setMask(oval_mask);
```

Quando si prova il documento (Controllo > Prova filmato), l'elemento mascherato rappresenta il canale alfa sfumato utilizzando la maschera.

NOTA

I livelli maschera non supportano l'applicazione dell'effetto maschera ai canali alfa. Per applicare una maschera è necessario utilizzare il codice ActionScript e la memorizzazione delle bitmap nella cache in fase di runtime.

Gestione di eventi clip filmato

I clip filmato possono rispondere a eventi utente, ad esempio clic del mouse o pressione dei tasti, e a eventi di sistema, come il caricamento iniziale di un clip filmato sullo stage.

ActionScript offre due modalità di gestione per gli eventi clip filmato: attraverso i metodi dei gestori di eventi e attraverso i gestori di eventi `onClipEvent()` e `on()`. Per ulteriori informazioni sulla gestione degli eventi clip filmato, vedere [Capitolo 9, “Gestione degli eventi”](#).

Assegnazione di una classe a un simbolo clip filmato

Con ActionScript 2.0 è possibile creare una classe che estenda il comportamento della classe MovieClip incorporata e quindi assegnare tale classe a un simbolo della libreria del clip filmato utilizzando la finestra di dialogo Proprietà del concatenamento. Ogni volta che viene creata un'istanza del clip filmato a cui la classe è assegnata, tale istanza assume le proprietà e i comportamenti definiti dalla classe assegnata. Per ulteriori informazioni su ActionScript 2.0, consultare il “[Esempio: creazione di classi personalizzate](#)” a pagina 237.

In una sottoclassificazione della classe MovieClip, è possibile fornire definizioni per i metodi e i gestori di eventi MovieClip incorporati, quali onEnterFrame e onRelease. Nella seguente procedura, viene creata una classe, denominata MoveRight, che estende la classe MovieClip e definisce un gestore onPress che sposta il filmato di 20 pixel a destra ogni volta che l'utente fa clic sul clip filmato. Nella seconda procedura, viene creato un simbolo clip filmato in un nuovo documento Flash (FLA) e si assegna la classe MoveRight al simbolo.

Per creare una sottoclassificazione di clip filmato:

1. Creare una nuova directory denominata BallTest.
2. Selezionare File > Nuovo, quindi scegliere File ActionScript dall'elenco dei tipi di documento per creare un nuovo file ActionScript.
3. Immettere il codice seguente nel file dello script:

```
// Classe MoveRight. Sposta il clip a destra di 20 pixel quando si fa
// clic su di esso
class MoveRight extends MovieClip {
    public function onPress() {
        this._x += 20;
    }
}
```

4. Salvare il documento come MoveRight.as nella directory BallTest.

Per assegnare la classe a un simbolo di clip filmato:

1. In Flash, selezionare File > Nuovo, selezionare un nuovo documento Flash dall'elenco di tipi di file e fare clic su OK.
2. Utilizzando lo strumento Ovale, disegnare un cerchio sullo stage.
3. Selezionare il cerchio e scegliere Elabora > Converti in simbolo.
4. Nella finestra di dialogo Converti in simbolo, selezionare Clip filmato come comportamento del simbolo e immettere ball_mc nella casella di testo Nome.

5. Selezionare Avanzato per visualizzare le opzioni relative al concatenamento, se non sono già visualizzate.
6. Selezionare Esporta per ActionScript e digitare **MoveRight** nella casella di testo Classe. Fare clic su OK.
7. Salvare il file come Ball.fla nella directory BallTest, la stessa directory contenente il file MoveRight.as.
8. Provare il documento Flash selezionando Controllo > Prova filmato.

Ogni volta che si fa clic sul clip filmato della pallina, esso si sposta di 20 pixel verso destra.

Se si creano proprietà del componente per una classe e si desidera che il clip filmato erediti tali proprietà, è necessario effettuare un'ulteriore operazione: dopo aver selezionato il simbolo del clip filmato nel pannello Libreria, scegliere Definizione componente dal menu a comparsa del pannello Libreria e immettere il nome della nuova classe nella casella Classe.

Inizializzazione delle proprietà delle classi

Nell'esempio presentato nella seconda procedura nella sezione “[Assegnazione di una classe a un simbolo clip filmato](#)”, è stata aggiunta manualmente l'istanza del simbolo Ball allo stage durante la creazione del codice. Come discusso in “[Aggiunta di parametri a clip filmato creati dinamicamente](#) a pagina 396”, è possibile assegnare parametri ai clip creati anche in fase di runtime utilizzando il parametro *initObject* di `attachMovie()` e `duplicateMovie()`. È possibile usare questa funzione per inizializzare le proprietà della classe in corso di assegnazione a un clip filmato.

Ad esempio, la classe `MoveRightDistance` seguente è una variazione della classe `MoveRight` (vedere “[Assegnazione di una classe a un simbolo clip filmato](#)” a pagina 412). La differenza è una nuova proprietà denominata `distance`, il cui valore determina di quanti pixel si sposta un clip filmato ogni volta che si fa clic su di esso.

Trasmissione di argomenti a una classe personalizzata:

1. Creare un nuovo documento ActionScript e salvarlo come **MoveRightDistance.as**.

2. Immettere il codice ActionScript seguente nella finestra Script:

```
// Classe MoveRightDistance -- sposta il clip a destra di 5 pixel ogni
// fotogramma.
class MoveRightDistance extends MovieClip {
    // La proprietà distance determina di quanti
    // pixel spostare il clip a ogni clic del mouse.
    var distance:Number;
    function onPress() {
        this._x += this.distance;
    }
}
```

3. Salvare il documento.

4. Creare un nuovo documento Flash e salvarlo come **MoveRightDistance.fla** nella stessa directory del file di classe.

5. Creare un simbolo del clip filmato contenente una forma vettoriale, come un ovale, e cancellare qualsiasi contenuto presente sullo stage.

Per questo esempio, nella libreria è necessario un solo simbolo del clip filmato.

6. Nel pannello Libreria, fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) sul simbolo e selezionare l'opzione Concatenamento dal menu di scelta rapida.

7. Assegnare al simbolo l'identificatore di concatenamento **Ball**.

8. Immettere **MoveRightDistance** nella casella di testo Classe AS 2.0.

9. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
this.attachMovie("Ball", "ball50_mc", 10, {distance:50});
this.attachMovie("Ball", "ball125_mc", 20, {distance:125});
```

Questo codice crea due nuove istanze del simbolo nella linea temporale principale del file SWF. La prima istanza, denominata **ball50_mc**, si sposta di 50 pixel ogni volta che l'utente fa clic su di essa; la seconda, denominata **ball125_mc**, si sposta di 125 pixel ogni volta che l'utente fa clic su di essa.

10. Selezionare Controllo > Prova filmato per provare il file SWF.

Operazioni con il testo e le stringhe

12

Molte delle applicazioni, presentazioni o immagini create mediante Macromedia Flash Professional 8 o Macromedia Flash Basic 8 includono porzioni di testo. È possibile utilizzare molti tipi di testo, ad esempio, testo statico nei layout o testo dinamico per le porzioni di testo più estese. È possibile, inoltre, utilizzare testo di input per acquisire l'input dell'utente e testo in un'immagine per creare la struttura dello sfondo. I campi di testo possono essere creati con lo strumento di creazione di Flash o mediante ActionScript.

Per visualizzare il testo è possibile, tra l'altro, manipolare le stringhe con codice prima che siano caricate e visualizzate sullo stage in fase di runtime. Le modalità d'uso delle stringhe in un'applicazione sono diverse; ad esempio, è possibile inviarle a un server, recuperare una risposta, analizzare le stringhe in un array oppure convalidare le stringhe immesse da un utente in un campo di testo.

In questo capitolo vengono descritti i vari modi di utilizzare testo e stringhe nelle applicazioni e, in particolare, l'uso del codice per manipolare il testo.

L'elenco di seguito descrive la terminologia utilizzata in questo capitolo.

Alias A differenza dell'antialiasing, il testo alias non utilizza variazioni di colore in modo che i bordi dei caratteri visualizzati sullo schermo appaiano meno irregolari (vedere descrizione successiva).

Antialiasing Consente di smussare il testo in modo che i bordi dei caratteri visualizzati sullo schermo appaiano meno irregolari. In Flash l'opzione Antialiasing allinea i contorni del testo lungo i limiti di pixel in modo da rendere il testo più leggibile, ed è particolarmente efficace per il rendering dei caratteri di piccole dimensioni.

Caratteri I caratteri sono lettere, numeri e segni di punteggiatura che combinati costituiscono le stringhe.

Caratteri di dispositivo Caratteri speciali in Flash non incorporati in un file SWF. Flash Player utilizza invece i caratteri disponibili sul computer locale che più assomigliano ai caratteri dispositivo. Poiché i profili del carattere non sono incorporati, le dimensioni di un file SWF sono inferiori rispetto all'utilizzo dei profili del carattere. Tuttavia, poiché i caratteri dispositivo non sono incorporati, sui sistemi che non hanno installato un tipo di carattere corrispondente al carattere dispositivo, il testo creato potrebbe risultare diverso da quello previsto. Flash include tre tipi di carattere dispositivo: `_sans` (simile a Helvetica o Arial), `_serif` (simile a Times Roman) e `_typewriter` (simile a Courier).

Caratteri Insiemi di caratteri con aspetto, stile e dimensioni simili.

Stringa Una sequenza di caratteri.

Testo Una serie di una o più stringhe che possono essere visualizzate in un campo di testo, o in un componente dell'interfaccia utente.

Campi di testo Elementi visivi sullo stage che consentono di visualizzare testo per un utente. Analogamente a un campo di testo di input o a un controllo di form area di testo in HTML, Flash consente l'impostazione dei campi di testo come modificabili (sola lettura) e la formattazione HTML, abilita il supporto multiriga, la mascheratura della password o l'applicazione di un foglio di stile CSS al testo formattato HTML.

Formattazione di testo È possibile applicare la formattazione a un campo di testo, oppure a certi caratteri all'interno di un campo di testo. Alcuni esempi delle opzioni di formattazione possibili sono: allineamento, rientri, grassetto, colore, dimensioni del carattere, larghezza dei margini, corsivo e spaziatura tra le lettere.

Per ulteriori informazioni sul testo, consultare i seguenti argomenti:

Informazioni sui campi di testo	417
Uso della classe TextField	418
Informazioni sul caricamento di testo e variabili nei campi di testo	427
Uso di caratteri	433
Informazioni sul rendering dei caratteri e sul testo con antialiasing	443
Informazioni sul layout e sulla formattazione del testo	452
Formattazione del testo con gli stili CSS	461
Creazione di un oggetto foglio di stile	463
Uso di un testo in formato HTML	475
Esempio: Creazione di testo scorrevole	491

Informazioni sui campi di testo

Un campo di testo dinamico o di input è un oggetto `TextField`, ovvero un'istanza della classe `TextField`. Quando si inserisce un campo di testo nell'ambiente di creazione, è possibile assegnarvi un nome di istanza nella finestra di ispezione Proprietà. Il nome dell'istanza può essere utilizzato nelle istruzioni ActionScript per impostare, modificare e formattare il campo di testo e il relativo contenuto utilizzando le classi `TextField` e `TextFormat`.

L'interfaccia utente e ActionScript consentono la creazione di diversi tipi di campi di testo. In Flash è possibile creare i tipi di campi di testo seguenti:

Testo statico Utilizzare il testo statico per visualizzare caratteri che non richiedono modifiche e per piccole quantità di testo oppure per visualizzare caratteri speciali che non sono disponibili nella maggior parte dei computer. Per visualizzare i caratteri non comuni, è inoltre possibile incorporarli per i campi di testo dinamici.

Testo dinamico Utilizzare il testo dinamico quando è necessario visualizzare caratteri che vengono aggiornati o modificati in fase di runtime. Nei campi di testo dinamici è anche possibile caricare del testo.

Testo di input Utilizzare i campi di testo di input quando è necessario acquisire l'input dell'utente. In questi campi di testo, infatti, l'utente può digitare.

Componenti di testo I componenti `TextArea` o `TextInput` consentono di visualizzare o acquisire testo nelle applicazioni. Il componente `TextArea` è simile a un campo di testo dinamico con barre di scorrimento incorporate, mentre il componente `TextInput` è simile a un campo di testo di input. Entrambi i componenti dispongono di funzionalità aggiuntive rispetto ai campi di testo equivalenti, ma causano l'aumento delle dimensioni dei file nell'applicazione.



Tutti i campi di testo supportano la codifica Unicode. Per informazioni su Unicode, vedere “[Informazioni sulle stringhe e sulla classe String](#)” a pagina 492.

I metodi della classe `TextField` consentono di impostare, selezionare e gestire il testo in un campo di testo dinamico o di input creato in fase di runtime o di creazione. Per ulteriori informazioni, vedere “[Uso della classe TextField](#)” a pagina 418. Per informazioni sul debug dei campi di testo in fase di runtime, vedere “[Informazioni sulla visualizzazione delle proprietà del campo di testo per il debug](#)” a pagina 788.

ActionScript fornisce inoltre svariati modi per formattare il testo in fase di runtime. La classe `TextFormat` consente di impostare la formattazione dei caratteri e dei paragrafi per gli oggetti `TextField` (vedere “[Uso della classe `TextFormat`](#) a pagina 458). Flash Player supporta inoltre un sottoinsieme di tag HTML utilizzabili per formattare il testo (vedere “[Uso di un testo in formato HTML](#)” a pagina 475). Flash Player 7 e le versioni successive supportano il tag HTML `img`, che consente di incorporare non solo immagini esterne, ma anche file SWF esterni e clip filmato che risiedono nella libreria. Vedere “[Tag per le immagini](#)” a pagina 479.

In Flash Player 7 e versioni successive, è possibile applicare stili CSS (Cascading Style Sheets) ai campi di testo che utilizzano la classe `TextField.StyleSheet`. Gli stili CSS possono essere utilizzati per specificare lo stile dei tag HTML incorporati, definire nuovi tag di formattazione o applicare stili. Per ulteriori informazioni sull'uso di CSS, vedere “[Formattazione del testo con gli stili CSS](#)” a pagina 461.

È possibile inoltre assegnare testo in formato HTML, che può eventualmente utilizzare gli stili CSS, direttamente a un campo di testo. In Flash Player 7 e versioni successive, il testo HTML assegnato a un campo di testo può includere contenuti multimediali incorporati, ad esempio clip filmato, file SWF e JPEG. In Flash Player 8, è inoltre possibile caricare dinamicamente immagini PNG, GIF e JPEG *progressivo* (Flash Player 7 non supporta il formato di immagini JPEG progressivo). Il testo viene disposto intorno al contenuto multimediale incorporato, in modo analogo al modo in cui in un browser il testo si dispone intorno al contenuto multimediale incorporato in un documento HTML. Per ulteriori informazioni, vedere “[Tag per le immagini](#)” a pagina 479.

Per ulteriori informazioni sulla terminologia che mette a confronto testo, stringhe e altri elementi, vedere l'introduzione di questo capitolo, “[Operazioni con il testo e le stringhe](#)” a pagina 415.

Uso della classe `TextField`

La classe `TextField` rappresenta qualunque campo di testo dinamico o di input creato mediante lo strumento Testo in Flash. È possibile utilizzare i metodi e le proprietà di questa classe per controllare i campi di testo in fase di runtime. Gli oggetti `TextField` supportano le stesse proprietà degli oggetti `MovieClip`, ad eccezione delle proprietà `_currentframe`, `_droptarget`, `_framesloaded` e `_totalframes`. È possibile ottenere e impostare le proprietà e richiamare in modo dinamico i metodi per i campi di testo.

Per controllare un campo di testo dinamico o di input utilizzando ActionScript, assegnarvi un nome di istanza nella finestra di ispezione Proprietà. Quindi, è possibile utilizzare il nome di istanza come riferimento del campo di testo e utilizzare i metodi e le proprietà della classe `TextField` per controllare il contenuto o l'aspetto di base del campo di testo.

Inoltre, è possibile creare oggetti TextField in fase di runtime e assegnarvi nomi di istanze mediante il metodo MovieClip.createTextField(). Per ulteriori informazioni, vedere [“Creazione di campi di testo in fase di runtime” a pagina 422](#).

Per ulteriori informazioni sull'uso della classe TextField, consultare i seguenti argomenti:

- [“Assegnazione di testo a un campo di testo in fase di runtime” a pagina 419](#)
- [“Informazioni sui nomi di istanze e di variabili dei campi di testo” a pagina 421](#)

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano textfieldsA.fla e textfieldsB.fla, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields*.

Assegnazione di testo a un campo di testo in fase di runtime

Durante la creazione di applicazioni con Flash, può essere necessario caricare testo da un'origine esterna, quale un file di testo, un file XML o un servizio Web remoto. Flash offre un alto grado di controllo sulla creazione e sulla visualizzazione di testo sullo stage, quale il supporto di testo formattato in HTML, testo normale, testo formattato in XML e fogli di stile esterni. In alternativa è possibile utilizzare ActionScript per la definizione di un foglio di stile.

Per assegnare testo a un campo di testo, utilizzare la proprietà `TextField.text` o `TextField.htmlText`. In alternativa, se nella finestra di ispezione Proprietà è stato inserito un valore nella variabile del campo di testo, è possibile assegnare un valore al campo di testo mediante la creazione di una variabile con il nome specificato. Se nel documento Flash si utilizza la versione 2 di Macromedia Components Architecture è anche possibile assegnare valori mediante la creazione di associazioni tra componenti.

L'esercizio seguente assegna testo a un campo di testo in fase di runtime:

Per assegnare testo a un campo di testo in fase di runtime:

1. Utilizzare lo strumento Testo per creare un campo di testo sullo stage.
2. Con il campo di testo selezionato, nella finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà), selezionare Testo di input dal menu a comparsa e immettere `headline_txt` nella casella di testo Nome istanza.

I nomi delle istanze possono essere costituiti solo da lettere, numeri, caratteri di sottolineatura (_) e simboli di dollaro (\$).

3. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
4. Immettere il codice seguente nel pannello Azioni:

```
headline_txt.text = "New articles available on Developer Center";
```

5. Selezionare Controllo > Prova filmato per provare il documento Flash.

È inoltre possibile creare un campo di testo con ActionScript e quindi assegnarvi del testo.

Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,  
300, 20);  
headline_txt.text = "New articles available on Developer Center";
```

Questo codice crea un nuovo campo di testo con il nome di istanza headline_txt. Il campo di testo viene creato alla successiva profondità maggiore, in corrispondenza delle coordinate *x* e *y* 100, 100, con un campo di testo di 200 pixel di larghezza e di 20 pixel di altezza. Quando si prova il file SWF, selezionando Controllo > Prova filmato, sullo stage viene visualizzato il testo "New articles available on Developer Center" (Nuovi articoli disponibili su Developer Center).

Per creare un campo di testo formattato in HTML:

Per abilitare la formattazione in HTML per il campo di testo, effettuare una delle due operazioni seguenti:

- Selezionare un campo di testo e fare clic sul pulsante Rendi il testo come HTML nella finestra di ispezione Proprietà.
- Utilizzando ActionScript impostare la proprietà `html` del campo di testo su `true` (vedere l'esempio di codice successivo).

Per applicare la formattazione in HTML a un campo di testo utilizzando ActionScript, nel fotogramma 1 della linea temporale immettere il seguente ActionScript:

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,  
300, 20);  
headline_txt.html = true;  
headline_txt.htmlText = "New articles available on <i>Developer Center</i>.";
```

Il codice precedente crea un nuovo campo di testo in modo dinamico, abilita la formattazione in HTML e visualizza sullo stage il testo “New articles available on Developer Center”, con le parole “Developer Center” in corsivo.

ATTENZIONE

Quando sullo stage si utilizza testo (non componenti) formattato in HTML, è necessario assegnare il testo alla proprietà `htmlText` del campo di testo e non alla proprietà `text`.

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano `textfieldsA.fla` e `textfieldsB.fla`, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields*.

Informazioni sui nomi di istanze e di variabili dei campi di testo

Nella casella di testo Nome istanza della finestra di ispezione Proprietà è necessario assegnare un nome di istanza a un campo di testo per richiamare i metodi e per ottenere e impostare le proprietà per quel campo di testo.

Nella casella di testo Var della finestra di ispezione Proprietà è possibile assegnare il nome di una variabile a un campo di testo di input o dinamico e quindi assegnare i valori desiderati alla variabile. Questa funzionalità obsoleta può essere utilizzata quando si creano applicazioni per le versioni precedenti di Flash Player (ad esempio Flash Player 4). Se si impostano come destinazione lettori più recenti, fare riferimento al testo contenuto in un campo di testo utilizzando il relativo nome di istanza e ActionScript.

Fare attenzione a non confondere il nome dell'istanza del campo di testo con il nome della relativa variabile. Il nome della variabile del campo di testo rappresenta un riferimento al testo contenuto in tale campo e non un riferimento a un oggetto.

Ad esempio, se si assegna il nome di variabile `myTextVar` a un campo di testo, è possibile impostare il contenuto del campo di testo utilizzando il codice seguente:

```
var myTextVar:String = "This is what will appear in the text field";
```

Tuttavia, non è possibile utilizzare il nome della variabile `myTextVar` per impostare la proprietà `text` del campo di testo. A questo scopo occorre fare ricorso al nome dell'istanza, come nel codice seguente:

```
//Tale soluzione non è valida.  
myTextVar.text = "A text field variable is not an object reference";  
  
// Funziona per il campo di testo di input con nome di istanza "myField".  
myField.text = "This sets the text property of the myField object";
```

Utilizzare la proprietà `TextField.text` per controllare il contenuto di un campo di testo, a meno che non si utilizzi una versione di Flash Player che non supporta la classe `TextField`. In questo modo infatti si riducono le possibilità di conflitto tra i nomi delle variabili e quindi il rischio di comportamenti imprevisti in fase di runtime.

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano `textfieldsA.fla` e `textfieldsB.fla`, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields*.

Creazione di campi di testo in fase di runtime

È possibile utilizzare il metodo `createTextField()` della classe `MovieClip` per creare un campo di testo vuoto sullo stage in fase di runtime. Il nuovo campo di testo viene associato alla linea temporale del clip filmato che chiama il metodo.

Per creare un campo di testo in modo dinamico tramite ActionScript:

1. Selezionare File > Nuovo, quindi Documento Flash per creare un nuovo file FLA.
2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:
`this.createTextField("test_txt", 10, 0, 0, 300, 100);`
Questo codice crea un campo di testo di 300 x 100 pixel denominato `test_txt` con una posizione (0, 0) e una profondità (z-order) di 10:

- 3.** È possibile accedere ai metodi e alle proprietà del nuovo campo di testo utilizzando il nome di istanza specificato nel primo parametro del metodo `createTextField()`.

Ad esempio, il codice seguente crea un nuovo campo di testo denominato `test_txt`, quindi ne modifica le proprietà in modo da ottenere un campo di testo multiriga con ritorno a capo e con la capacità di espandersi per adattarsi al testo immesso. Infine, assegna un testo utilizzando la proprietà `text` del campo di testo:

```
test_txt.multiline = true;  
test_txt.wordWrap = true;  
test_txt.autoSize = "left";  
test_txt.text = "Create new text fields with the  
MovieClip.createTextField() method.";
```

- 4.** Selezionare Controllo > Prova filmato per visualizzare il campo di testo.

Il testo viene creato in fase di runtime e visualizzato sullo stage.

È possibile utilizzare il metodo `TextField.removeTextField()` per rimuovere un campo di testo creato con `createTextField()`. Il metodo `removeTextField()` non può essere utilizzato in un campo di testo posizionato dalla linea temporale durante la creazione.

Per ulteriori informazioni, vedere `createTextField` (`MovieClip.createTextField` method) e `removeTextField` (`TextField.removeTextField` method) nella *Guida di riferimento di ActionScript 2.0*.

NOTA

Alcune proprietà `TextField`, come `_rotation`, non sono disponibili quando si creano campi di testo in fase di runtime. Un campo di testo può essere ruotato solo se utilizza caratteri incorporati. Vedere ["Per incorporare un simbolo di carattere:" a pagina 436](#).

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano `textfieldsA.fla` e `textfieldsB.fla`, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a `unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields`.
- Su Macintosh, accedere a `Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields`.

Informazioni sulla modifica dei campi di testo

È possibile modificare in molti modi i campi di testo creati in un file FLA: un campo di testo può essere modificato a condizione di assegnarvi un nome di istanza nella finestra di ispezione Proprietà oppure mediante il codice, se il campo è stato creato utilizzando il codice. Nel semplice esempio seguente viene creato un campo di testo, viene assegnata al campo una stringa di testo e viene modificata la proprietà border del campo:

```
this.createTextField("pigeon_txt", this.getNextHighestDepth(), 100, 100,  
200, 20);  
pigeon_txt.text = "I like seeds";  
pigeon_txt.border = true;
```

Per un elenco completo delle proprietà della classe TextField, consultare la *Guida di riferimento di ActionScript 2.0*.

Per alcuni esempi relativi alla modifica dei campi di testo, consultare le sezioni seguenti:

- “Modifica della posizione di un campo di testo” a pagina 424
- “Modifica delle dimensioni di un campo di testo in fase di runtime” a pagina 425

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano textfieldsA.fla e textfieldsB.fla, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields*.

Modifica della posizione di un campo di testo

La posizione di un campo di testo viene modificata sullo stage in fase di runtime, mediante l'impostazione di nuovi valori per le proprietà _x e _y, come illustrato nell'esempio seguente.

Per riposizionare un campo di testo tramite ActionScript:

1. Creare un nuovo file FLA e salvarlo come **positionText.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 0, 0, 300, 200);  
my_txt.border = true;  
my_txt.text = "Hello world";  
my_txt._x = (Stage.width - my_txt._width) / 2;  
my_txt._y = (Stage.height - my_txt._height) / 2;
```
3. Salvare il documento Flash e selezionare Controllo > Prova filmato per visualizzare il campo di testo centrato sullo stage.

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano textfieldsA.fla e textfieldsB.fla, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/TextFields*.

Modifica delle dimensioni di un campo di testo in fase di runtime

Potrebbe essere necessario ottenere o impostare in modo dinamico le dimensioni di un campo di testo in fase di runtime, invece che nell'ambiente di creazione. Nell'esempio seguente viene creato un campo di testo su una linea temporale e ne vengono impostate le dimensioni iniziali su 100 pixel di larghezza per 21 pixel di altezza. Successivamente, il campo di testo viene ridimensionato su 300 pixel di larghezza per 200 pixel di altezza e quindi viene riposizionato al centro dello stage.

Per ridimensionare un campo di testo tramite ActionScript:

1. Creare un nuovo documento Flash e salvarlo come **resizeText.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 0, 0, 100, 21);
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Hello world";
my_txt.wordWrap = true;
my_txt._width = 300;
my_txt._height = 200;
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per visualizzare i risultati nell'ambiente di creazione.

Nell'esempio precedente è stato ridimensionato un campo di testo creato in modo dinamico mediante l'impostazione su 300 x 200 pixel in fase di runtime; quando tuttavia si carica contenuto da un sito Web esterno, e non si è certi della quantità di contenuto restituito, questa tecnica potrebbe non rivelarsi adatta alle proprie esigenze. Per ovviare a questo possibile inconveniente, Flash dispone del metodo `TextField.autoSize`, che può essere utilizzato per ridimensionare automaticamente un campo di testo in modo che si adatti al contenuto.

Nell'esempio seguente viene dimostrato come utilizzare la proprietà `TextField.autoSize` per ridimensionare il campo di testo dopo che è stato inserito del testo.

Per ridimensionare automaticamente dei campi di testo in base al contenuto:

1. Creare un nuovo documento Flash e salvarlo come **resizeTextAuto.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("my_txt", 10, 10, 10, 160, 120);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
    enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
    ut aliquip ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum.";
my_txt.wordWrap = true;
```

NOTA

Se il codice viene incollato direttamente nel pannello Azioni da alcune versioni della Guida in linea di Flash, nella lunga stringa di testo si possono trovare interruzioni di riga. In tal caso il codice non viene compilato; per ovviare al problema, selezionare Caratteri nascosti dal menu a comparsa del pannello Azioni, quindi eliminare i caratteri di interruzione di riga nella lunga stringa di testo.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per visualizzare il documento Flash nell'ambiente di creazione.

Flash ridimensiona il campo di testo verticalmente, in modo che tutto il contenuto possa essere visualizzato senza essere ritagliato in base ai limiti del campo di testo. Se si imposta la proprietà `my_txt.wordWrap` su `false`, il campo di testo viene ridimensionato orizzontalmente in modo da contenere il testo.

Per applicare un'altezza massima al campo di testo ridimensionato automaticamente, in modo che non superi i limiti dello stage, è possibile utilizzare il codice seguente.

```
if (my_txt._height > 160) {
    my_txt.autoSize = "none";
    my_txt._height = 160;
}
```

Per consentire agli utenti di visualizzare la parte di testo restante, è necessario aggiungere una funzione di scorrimento, quale ad esempio una barra di scorrimento, oppure è possibile scorrere il puntatore del mouse sul testo; spesso questo metodo è adatto durante le operazioni di test del codice.

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano textfieldsA.fla e textfieldsB.fla, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio*\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/TextFields.

Informazioni sul caricamento di testo e variabili nei campi di testo

Esistono vari modi per caricare testo in un documento Flash, ad esempio FlashVars, LoadVars, XML o i servizi Web, per citarne alcuni. Il metodo probabilmente più semplice per passare testo a un documento Flash consiste nell'utilizzare la proprietà FlashVars, che passa brevi stringhe di testo a un documento Flash mediante i tag `object` ed `embed` nel codice HTML utilizzato per incorporare il file SWF in una pagina HTML. Un altro metodo per caricare testo o variabili in un documento Flash è l'utilizzo della classe LoadVars, in grado di caricare da un file di testo grandi blocchi di testo o una serie di variabili in formato URL.

Come illustrato nell'esempio precedente in questa sezione, alcuni metodi di caricamento del testo in un file SWF sono meno complessi di altri. Se tuttavia si raccolgono dati da siti esterni, il formato dei dati che è necessario caricare potrebbe non essere tra le opzioni disponibili.

Ogni metodo di caricamento e/o invio di dati da e verso un file SWF ha lati positivi e negativi. XML, i servizi Web e Flash Remoting sono i più versatili per il caricamento di dati esterni, ma presentano la curva di apprendimento più impegnativa. Per informazioni su Flash Remoting, visitare il sito www.macromedia.com/support/flashremoting.

FlashVars e LoadVars sono molto più semplici, come illustrato in “[Uso di FlashVars per caricare e visualizzare testo](#)” a pagina 428 e in “[Uso di LoadVars per caricare e visualizzare testo](#)” a pagina 430, ma possono presentare maggiori limitazioni in relazione ai tipi e ai formati di dati che è possibile caricare. Inoltre, quando si inviano e si caricano dati, è necessario seguire alcune restrizioni sulla sicurezza. Per informazioni sulla sicurezza, consultare il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#) Per ulteriori informazioni sul caricamento dei dati esterni, consultare il [Capitolo 16, “Operazioni con i dati esterni”](#)

Nelle sezioni seguenti sono illustrati i diversi metodi di caricamento del testo e delle variabili nei documenti:

- “[Uso di FlashVars per caricare e visualizzare testo](#)” a pagina 428
- “[Uso di LoadVars per caricare e visualizzare testo](#)” a pagina 430
- “[Caricamento di variabili mediante LoadVars](#)” a pagina 431
- “[Caricamento e visualizzazione di testo da un documento XML](#)” a pagina 432

È possibile trovare file sorgente di esempio che illustrano come utilizzare i campi di testo con ActionScript. I file sorgente si chiamano loadText.fla e formattedText.fla, si trovano nella cartella Samples del disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/LoadText*.

Si trova inoltre un file sorgente che carica del testo e applica formattazione antialiasing, oltre a utilizzare la memorizzazione delle bitmap nella cache. Il file sorgente si chiama flashtype.fla nella cartella Samples sul disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType*.

Uso di FlashVars per caricare e visualizzare testo

L'uso di FlashVars è semplice, ma richiede la pubblicazione dei file SWF insieme ai documenti HTML. Si modifica il codice HTML generato e si includono le proprietà FlashVars nei tag object e embed. Il documento Flash può essere provato visualizzando il documento HTML modificato nel browser Web.

Per utilizzare FlashVars per passare variabili dal documento HTML al documento Flash:

1. Creare un nuovo documento Flash e salvarlo come **flashvars.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 10, 10, 100, 21);
my_txt.text = _level0.username;
```

3. Salvare il documento Flash e selezionare File > Pubblica per generare i file HTML e SWF.

NOTA

Un documento HTML viene pubblicato, per impostazione predefinita, nella stessa directory del file FLA. Se non viene pubblicato, selezionare File > Impostazioni pubblicazione, quindi scegliere la scheda Formati. Verificare di avere selezionato HTML.

4. Aprire il documento flashvars.html in un editor di testo o HTML.
5. Nel documento HTML, modificare il codice all'interno del tag object in modo che corrisponda a quanto indicato di seguito.

Il codice che è necessario aggiungere è in **grassetto**.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
       codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
       swflash.cab#version=8,0,0,0" width="550" height="400" id="flashvars"
       align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="flashvars.swf" />
<param name="FlashVars" value="username=Thomas" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="flashvars.swf" FlashVars="username=Thomas" quality="high"
       bgcolor="#ffffff" width="550" height="400" name="flashvars"
       align="middle" allowScriptAccess="sameDomain" type="application/x-
       shockwave-flash" pluginspage="http://www.macromedia.com/go/
       getflashplayer" />
</object>
```

6. Salvare le modifiche al documento HTML.
7. Aprire il documento HTML modificato in un browser Web.

Nel campo di testo creato in modo dinamico sullo stage, il file SWF visualizza il nome Thomas.

Per informazioni sulla sicurezza, consultare il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)

Uso di LoadVars per caricare e visualizzare testo

Per caricare contenuto in un file SWF, è inoltre possibile utilizzare la classe LoadVars che carica testo o variabili da un file esterno nello stesso server o anche contenuto da un server diverso. Nell'esempio seguente viene dimostrato come creare in modo dinamico un campo di testo e come compilarlo con il contenuto di un file di testo remoto.

Per utilizzare LoadVars per compilare un campo di testo con testo esterno:

1. Creare un nuovo documento Flash e salvarlo come **loadvarsText.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function (src:String):Void {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Unable to load external file.";
    }
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Il primo blocco di codice nel segmento precedente crea un nuovo campo di testo sullo stage, nel quale è possibile immettere testo multiriga e con ritorno a capo. Il secondo blocco di codice definisce un nuovo oggetto LoadVars, che viene utilizzato per caricare un file di testo (lorem.txt) da un server Web remoto e per visualizzarne il contenuto nel campo di testo my_txt creato in precedenza.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF. Dopo un breve intervallo, Flash visualizza il contenuto del file remoto nel campo di testo sullo stage.

Per informazioni sulla sicurezza, consultare il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)

Caricamento di variabili mediante LoadVars

La classe LoadVars consente inoltre di caricare variabili in un formato URL, in modo analogo al passaggio di variabili in una stringa di query in un browser Web. Nell'esempio seguente viene dimostrato come caricare un file di testo remoto in un file SWF e cone visualizzare le relative variabili, monthNames e dayNames.

Per caricare variabili da un file di testo mediante LoadVars:

1. Creare un nuovo documento Flash e salvarlo come **loadvarsVariables.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onLoad = function (success:Boolean):Void {
    if (success) {
        my_txt.text = "dayNames: " + lorem_lv.dayNames + "\n\n";
        my_txt.text += "monthNames: " + lorem_lv.monthNames;
    } else {
        my_txt.text = "Unable to load external file.";
    }
}
/* contents of params.txt:
   &monthNames=January,February,...&dayNames=Sunday,Monday,...
 */
lorem_lv.load("http://www.helpexamples.com/flash/params.txt");
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato dal menu principale.

L'uso del metodo `LoadVars.onLoad()` anziché di `LoadVars.onData()` fa sì che Flash analizzi le variabili e le crei all'interno dell'istanza dell'oggetto LoadVars. Il file di testo esterno contiene due variabili, `monthNames` e `dayNames`, che contengono entrambe delle stringhe.

Per informazioni sulla sicurezza, consultare il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)

Caricamento e visualizzazione di testo da un documento XML

L'uso di dati XML è un metodo diffuso per la distribuzione di contenuto su Internet, in parte perché si tratta di uno standard ampiamente accettato per l'organizzazione e l'analisi dei dati. Per questo motivo, il formato XML è una scelta eccellente per l'invio e la ricezione di dati da Flash, benché per il caricamento e la visualizzazione del testo presenti una curva di apprendimento leggermente più impegnativa rispetto all'utilizzo di LoadVars e di FlashVars.

Per caricare testo in Flash da un documento XML esterno:

1. Creare un nuovo documento Flash e salvarlo come **xmlReviews.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var reviews_xml:XML = new XML();
reviews_xml.ignoreWhite = true;
reviews_xml.onLoad = function (success:Boolean):Void {
    if (success) {
        var childItems:Array = reviews_xml.firstChild.childNodes;
        for (var i:Number = 0; i < childItems.length; i++) {
            my_txt.text += childItems[i].firstChild.firstChild.nodeValue +
            "\n";
        }
    } else {
        my_txt.text = "Unable to load external file.";
    }
}
reviews_xml.load("http://www.helpexamples.com/flash/xml/reviews.xml");
```

Il primo blocco di codice nel segmento precedente crea un nuovo campo di testo sullo stage. Questo campo di testo viene utilizzato per visualizzare diverse parti del documento XML caricato successivamente. Il secondo blocco di codice gestisce la creazione di un oggetto XML che viene utilizzato per caricare il contenuto XML. Dopo aver completato il caricamento e l'analisi dei dati da parte di Flash, viene richiamato il gestore di eventi XML.onLoad() e viene visualizzato il contenuto del pacchetto XML nel campo di testo.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Flash visualizza l'output seguente nel campo di testo sullo stage:

```
Item 1  
Item 2  
...  
Item 8
```

Per informazioni sulla sicurezza, consultare il [Capitolo 17, “Nozioni fondamentali sulla sicurezza”](#)

Uso di caratteri

I caratteri sono insiemi di caratteri con aspetto, stile e dimensioni simili. Indipendentemente dal tipo di elemento creato con Flash Basic 8 o Flash Professional 8, il testo utilizzato nelle applicazioni Flash include probabilmente almeno uno o due tipi di caratteri. Se si creano animazioni e non si ha la certezza che nei sistemi degli utenti finali sia installato un carattere specifico, è necessario apprendere i fondamenti delle operazioni di incorporamento dei caratteri.

Nelle sezioni seguenti viene illustrato come incorporare caratteri, intere gamme di caratteri, caratteri condivisi e vengono descritte altre tecniche relative all'uso di caratteri in Flash 8.

Per ulteriori informazioni sui caratteri, consultare le seguenti sezioni:

- [“Incorporamento di caratteri” a pagina 434](#)
- [“Incorporamento di caratteri” a pagina 436](#)
- [“Creazione di set di caratteri personalizzati” a pagina 438](#)
- [“Uso dei metodi TextField con caratteri incorporati” a pagina 441](#)
- [“Informazioni sulla condivisione dei caratteri” a pagina 442](#)

L'esempio seguente illustra come aggiungere ed eliminare caratteri incorporati e set di caratteri in un documento Flash.

Per aggiungere ed eliminare caratteri incorporati e set di caratteri:

1. Creare un nuovo documento Flash e salvarlo come `embedding.fla`.
2. Con lo strumento Testo creare un campo di testo dinamico sullo stage.
3. Fare clic sul pulsante Incorpora per visualizzare la finestra di dialogo Incorporamento caratteri.

4. Selezionare un set di caratteri da incorporare facendo clic su di esso.

Per selezionare set di caratteri multipli, durante la selezione degli elementi con il puntatore tenere premuti i tasti Maiusc o Ctrl. Per selezionare un blocco di set di caratteri, selezionare il primo con il puntatore del mouse, tenere premuto il tasto Maiusc e fare clic su un altro set di caratteri. Con Maiusc vengono selezionati tutti i set di caratteri tra i due set di caratteri selezionati. Per selezionare set di caratteri multipli non consecutivi, tenere premuto il tasto Ctrl durante la selezione dei set di caratteri. È anche possibile selezionare velocemente set di caratteri multipli selezionando un set di caratteri con il mouse e, con il pulsante del mouse premuto, trascinare il mouse su set di caratteri multipli.

5. Per eliminare un set di caratteri aggiunto in precedenza, tenere premuto il tasto Ctrl e deselectonare il set di caratteri facendo clic su di esso.
6. Per eliminare tutti i set di caratteri selezionati e tutti i caratteri specificati nel campo di testo Includi questi caratteri, selezionare Non incorporare; in questo modo vengono eliminati tutti i caratteri o set di caratteri specificati in precedenza.



La selezione di Non incorporare nella finestra di dialogo Incorporamento caratteri elimina tutti i caratteri e set di caratteri incorporati specificati scelti in precedenza senza chiedere conferma.

Incorporamento di caratteri

Se si lavora con caratteri incorporati e si conoscono i caratteri necessari, è possibile ridurre le dimensioni del file incorporando solo i caratteri necessari, senza includere ulteriori profili di carattere non utilizzati. Per incorporare in un campo di testo solo alcuni caratteri e non un set di caratteri completo, utilizzare la finestra di dialogo Incorporamento caratteri per specificare i caratteri che si desidera incorporare.

Per incorporare caratteri specifici da utilizzare in un campo di testo:

1. Creare un nuovo documento Flash e salvarlo come charembbed.fla.
2. Con lo strumento Testo creare un campo di testo sullo stage e impostare il tipo di testo del campo come dinamico o di input.
3. Con il campo di testo ancora selezionato nello stage, fare clic su Incorpora nella finestra di ispezione Proprietà.

Viene visualizzata la finestra di dialogo Incorporamento caratteri che consente di impostare i set di caratteri che verranno incorporati nel documento Flash (oltre al numero di glifi per ciascun set di caratteri), di specificare caratteri specifici da incorporare e indica il numero totale di glifi che verranno incorporati per il campo di testo.

4. Nella casella di testo Includi questi caratteri, immettere la stringa **hello world**.

La finestra di dialogo indica che per il campo di testo verrà incorporato un totale di 8 glifi. Sebbene la stringa "hello world" contenga 11 caratteri, Flash incorpora glifi unici, quindi le lettere l e o vengono incorporate una sola volta, non due.

5. Fare clic su OK per applicare le modifiche e tornare al documento.
6. Con lo strumento Testo creare un nuovo campo di testo sullo stage.
7. Nella finestra di ispezione Proprietà selezionare Testo dinamico dal menu a comparsa Tipo testo.
8. Nel campo di testo sullo stage immettere la stringa **hello world**.
9. Fare clic sul pulsante Incorpora nella finestra di ispezione Proprietà per aprire nuovamente la finestra di dialogo Incorporamento caratteri.
10. Per compilare automaticamente la casella di testo Includi questi caratteri, fare clic su Riempimento automatico:

verrà visualizzata la stringa "hel0 wrd". Flash è in grado di determinare i caratteri unici nel campo di testo specificato, senza che l'utente indichi i caratteri che desidera incorporare.

SUGGERIMENTO

Flash è in grado di determinare automaticamente i caratteri da incorporare solo se il campo di testo contiene testo sullo stage; se il campo di testo viene compilato mediante ActionScript, è necessario specificare i caratteri da incorporare per il campo di testo.

11. Fare clic su OK.

Incorporamento di caratteri

Quando si incorporano caratteri, Flash memorizza tutte le relative informazioni nel file SWF, in modo da visualizzare correttamente il carattere anche se non è installato nel computer dell'utente. Se nel file FLA viene utilizzato un carattere non installato nel sistema dell'utente e questo carattere non viene incorporato nel file SWF, Flash Player seleziona automaticamente un carattere sostitutivo da utilizzare al posto del carattere mancante.

NOTA

È necessario incorporare un carattere solo se si utilizzano campi di testo dinamico o di input; in un campo di testo statico, infatti, questa operazione non è richiesta.

Per incorporare un simbolo di carattere:

1. Selezionare Finestra > Libreria per aprire la libreria di file FLA corrente.
Aprire la libreria a cui si desidera aggiungere un simbolo di carattere.
2. Selezionare Nuovo carattere dal menu a comparsa della libreria, nell'angolo superiore destro del pannello Libreria.
3. Nella finestra di dialogo Proprietà simbolo carattere, digitare un nome per il simbolo di carattere nella casella di testo Nome.
4. Selezionare un carattere dal menu Carattere oppure digitare il nome di un carattere nella relativa casella di testo.
5. Per applicare uno stile al carattere, selezionare Grassetto, Corsivo o Testo alias.
6. Inserire la dimensione di carattere da incorporare, quindi fare clic su OK per applicare le modifiche e tornare al documento.

Il carattere viene visualizzato nella libreria nel documento corrente.

Dopo aver incorporato un carattere nella libreria, è possibile utilizzarlo con un campo di testo sullo stage.

Per utilizzare un simbolo di carattere incorporato nel documento Flash:

1. Per incorporare un carattere nella Libreria, seguire le procedure riportate in [“Incorporamento di caratteri” a pagina 436](#).
2. Usare lo strumento Testo per creare un campo di testo sullo stage.
3. Digitare del testo nel campo di testo.
4. Selezionare il campo di testo e aprire la finestra di ispezione Proprietà.
 - a. Impostare il campo di testo su una riga singola.
 - b. Selezionare il nome del carattere incorporato utilizzando il menu a discesa Carattere.

I caratteri incorporati presentano un asterisco (*) dopo il nome di carattere.

5. Fare clic sul pulsante Incorpora nella finestra di ispezione Proprietà per aprire la finestra di dialogo Incorporamento caratteri.

La finestra di dialogo Incorporamento caratteri consente di selezionare i singoli caratteri o i set di caratteri da incorporare per il campo di testo selezionato. Per specificare i caratteri da incorporare, digitarli caratteri nella casella di testo della finestra di dialogo oppure compilare automaticamente il campo di testo, mediante il pulsante Riempimento automatico, con i caratteri univoci che si trovano attualmente nel campo di testo. Se non si conoscono esattamente i caratteri necessari, ad esempio se il testo viene caricato da un file esterno o da un servizio Web, è possibile selezionare interi set di caratteri da incorporare, ad esempio Maiuscolo [A..Z], Minuscolo [a..z], Numeri [0..9], Punteggiatura [!@#%...] e set di caratteri per lingue diverse.

NOTA

Ogni set di caratteri selezionato aumenta le dimensioni del file SWF finale, poiché Flash deve memorizzare tutte le informazioni sui caratteri per ogni set di caratteri utilizzato.

6. Selezionare i singoli caratteri o i set di caratteri che si desidera incorporare e quindi fare clic su OK per applicare le modifiche e tornare al documento.
7. Selezionare Controllo > Prova filmato per provare il documento Flash nell'ambiente di creazione.

Nel campo di testo sullo stage viene visualizzato il carattere incorporato. Per verificare che il carattere sia incorporato nel modo corretto, potrebbe essere necessario effettuare la prova su un computer separato, in cui non sia installato il carattere incorporato.

In alternativa, è possibile impostare la proprietà `TextField._alpha` o `TextField._rotation` del campo di testo con caratteri incorporati; queste proprietà, infatti, funzionano solo con quel tipo di caratteri (vedere le procedure seguenti).

8. Chiudere il file SWF e tornare allo strumento di creazione.
9. Selezionare il campo di testo sullo stage e aprire la finestra di ispezione Proprietà.
 - a. Selezionare Testo dinamico dal menu a comparsa Tipo testo per il campo di testo.
 - b. Digitare `font_txt` nella casella di testo Nome istanza.
10. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
font_txt._rotation = 45;
```

11. Selezionare di nuovo Controllo > Prova filmato per visualizzare le modifiche nell'ambiente di creazione.

Il carattere incorporato viene ruotato di 45 gradi in senso orario ed è ancora possibile visualizzare il testo poiché è incorporato nel file SWF.

ATTENZIONE

Se non si incorpora un carattere nel documento Flash, e Flash Player sceglie automaticamente un carattere sostitutivo sul computer dell'utente, la proprietà TextField.font restituisce il carattere originale utilizzato nel file FLA, non il nome del carattere sostituito.

NOTA

Se nei campi di testo si utilizzano caratteri incorporati con stili diversi, è necessario incorporare lo stile da utilizzare. Ad esempio, se si utilizza un carattere incorporato chiamato Times e si desidera avere una parola in corsivo, è necessario incorporare i profili di carattere normale e corsivo, in caso contrario il testo non viene visualizzato nel campo di testo.

Creazione di set di caratteri personalizzati

Oltre all'utilizzo dei set di caratteri predefiniti di Flash, è possibile creare set di caratteri personalizzati e aggiungerli alla finestra di dialogo Incorporamento caratteri. Ad esempio, può essere necessario che alcuni campi includano Latino esteso, per il supporto ai caratteri accentati; tuttavia potrebbero non essere necessari numeri e punteggiatura, oppure sono necessari solo caratteri maiuscoli. È possibile creare un set di caratteri personalizzato contenente solo i caratteri necessari, senza dover incorporare il set di caratteri completo. In tal modo è possibile mantenere ridotte le dimensioni del file SWF, poiché non vengono memorizzate ulteriori informazioni sui caratteri non necessari.

Per creare un set di caratteri personalizzato è necessario modificare il file UnicodeTable.xml file, situato nella directory C:\Programmi\Macromedia\Flash 8\<lingua>\FirstRun\FontEmbedding\. Il file definisce i set di caratteri predefiniti, oltre agli intervalli di caratteri e ai caratteri in essi contenuti.

Prima di creare un set di caratteri personalizzato, è necessario comprendere la struttura XML necessaria alla creazione. I seguenti nodi XML definiscono il set di caratteri Maiuscolo [A..Z]:

```
<glyphRange name="Maiuscolo [A..Z]" id="1">
  <range min="0x0020" max ="0x0020" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

Si noti che il nodo `glyphRange` contiene `name`, Maiuscolo [A..Z] e `id`. Un nodo `glyphRange` può avere tutti i nodi secondari *range* necessari. Un range, o intervallo, può essere un singolo carattere, come `0x0020` (il carattere spazio), presente nel precedente frammento, oppure un intervallo di caratteri, come nell'intervallo del secondo nodo secondario. Per incorporare un singolo carattere, impostare il valore `min` e il valore `max` sul medesimo valore di carattere unicode.

Un altro esempio di nodo XML `glypjRange` è il nodo `Numeri [0..9]`:

```
<glyphRange name="Numeri [0..9]" id="3">
  <range min="0x0030" max ="0x0039" />
  <range min="0x002E" max ="0x002E" />
</glyphRange>
```

Questo intervallo di caratteri comprende i valori Unicode da `0x0030` (zero) a `0x0039` (9), oltre a `0x002E` (.).

Prima di creare un set di caratteri personalizzato, è necessario conoscere i caratteri e i valori Unicode corrispondenti. I valori Unicode si trovano nel sito Web Unicode Standards, all'indirizzo www.unicode.org, contenente le tabelle dei codici di caratteri Unicode per decine di lingue.

ATTENZIONE

Per aggiungere set di caratteri personalizzati è necessario modificare un file XML situato nella cartella di installazione di Flash. Prima della modifica, effettuare una copia di backup, nel caso si desiderasse ripristinare la tabella Unicode originale.

ATTENZIONE

Macromedia consiglia di non modificare i set di caratteri esistenti installati con Flash e di creare i propri set di caratteri personalizzati contenenti i caratteri e la punteggiatura necessari alle applicazioni.

Per creare e utilizzare un set di caratteri personalizzati:

1. Con un editor XML o di testo, quale Blocco note o *TextEdit*, aprire il documento `UnicodeTable.xml`, situato in `<directory di installazione di Flash>\<lingua>\FirstRun\FontEmbedding\`.

NOTA

Ricordare di effettuare un backup del documento, nel caso si desideri ripristinare il file originale installato con Flash.

- 2.** Scorrere nella parte finale del documento XML e aggiungere il seguente codice XML appena prima del nodo di chiusura </fontEmbeddingTable>:

```
<glyphRange name="Maiuscolo e Numeri [A..Z,0..9]" id="100">
  <range min="0x0020" max ="0x0020" />
  <range min="0x002E" max ="0x002E" />
  <range min="0x0030" max ="0x0039" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

- 3.** Salvare le modifiche apportate a UnicodeTable.xml.

Se Flash è aperto, prima di utilizzare il nuovo set di caratteri è necessario riavviare l'applicazione.

- 4.** Aprire o riavviare Flash, quindi creare un nuovo documento Flash.

- 5.** Con lo strumento Testo aggiungere una nuova istanza TextField sullo stage.

- 6.** Nella finestra di ispezione Proprietà impostare Tipo testo di TextField su Testo dinamico, quindi fare clic su Modifica le opzioni per i caratteri per aprire la finestra di dialogo Incorporamento caratteri.

- 7.** Scorrere in fondo alla finestra di dialogo Incorporamento caratteri e selezionare il nuovo set di caratteri personalizzato, Maiuscolo e Numeri [A..Z,0..9] (38 glifi).

- 8.** Selezionare qualsiasi altro set di caratteri e fare clic su OK.

Selezionando il set di caratteri personalizzato, Maiuscolo e Numeri [A..Z,0..9], e il set di caratteri predefinito Maiuscolo [A..Z] o Numeri [0..9], il numero di glifi incorporati non cambia, poiché tutti i caratteri maiuscoli sono inclusi nel set di caratteri personalizzato e Flash non include caratteri duplicati, mantenendo al minimo le dimensioni del file.

Selezionando il set di caratteri Punteggiatura, contenente 52 glifi, e il set di caratteri personalizzato, contenente 38 glifi, Flash memorizza informazioni su 88 glifi e non su 90, poiché due caratteri, spazio e punto, sono ripetuti e sono già inclusi nel set di caratteri personalizzato.

SUGGERIMENTO

La posizione di un set di caratteri nella finestra di dialogo Incorporamento caratteri dipende dalla sua posizione nel documento XML. È possibile modificare l'ordine dei set di caratteri, predefiniti e personalizzati, spostando i pacchetti <glyphRange> nel file XML.

Uso dei metodi TextField con caratteri incorporati

I metodi della classe TextField forniscono utili funzionalità per le applicazioni; ad esempio, è possibile controllare lo spessore di un campo di testo mediante ActionScript, come illustrato nell'esempio seguente.

Per impostare lo spessore di un campo di testo tramite ActionScript:

1. Creare un nuovo documento Flash e salvarlo come **textfieldThickness.fla**.
2. Aprire il pannello Libreria e selezionare Nuovo carattere dal menu a comparsa nell'angolo superiore destro del pannello.

Viene visualizzata la finestra di dialogo Proprietà del simbolo di carattere. In questa finestra di dialogo è possibile selezionare un carattere da incorporare nel file SWF (compresi uno stile e le dimensioni del carattere). È inoltre possibile assegnare un nome di carattere visualizzato nella libreria del documento e nel menu a discesa Carattere della finestra di ispezione Proprietà (se sullo stage è selezionato un campo di testo).

 - a. Selezionare il carattere Times New Roman dal menu a discesa Carattere.
 - b. Verificare che le opzioni Grassetto e Corsivo siano deselezionate.
 - c. Impostare le dimensioni su 30 pixel.
 - d. Immettere il nome di carattere **Times (embedded)**.
 - e. Fare clic su OK.
3. Nella libreria, fare clic con il pulsante destro del mouse sul simbolo del carattere e selezionare Concatenamento dal menu di scelta rapida.

Viene aperta la finestra di dialogo Proprietà del concatenamento.
4. Selezionare le opzioni Esporta per ActionScript ed Esporta nel primo fotogramma, quindi fare clic su OK.
5. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// 1
this.createTextField("thickness_txt", 10, 0, 0, Stage.width, 22);
this.createTextField("lorem_txt", 20, 0, 20, Stage.width, 0);
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";
lorem_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est laborum.";
lorem_txt.wordWrap = true;
```

```

// 2
var style_fmt:TextFormat = new TextFormat();
style_fmt.font = "Times (embedded)";
style_fmt.size = 30;
lorem_txt.setTextFormat(style_fmt);

// 3
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
    // I valori per TextField.thickness sono compresi tra -200 e +200.
    lorem_txt.thickness = Math.round(_xmouse * (400 / Stage.width) - 200);
    thickness_txt.text = "TextField.thickness = " + lorem_txt.thickness;
};
Mouse.addListener(mouseListener);

```

Il primo blocco di codice crea due campi di testo, `thickness_txt` e `lorem_txt`, e li posiziona sullo stage. Il campo di testo `lorem_txt` imposta la proprietà `embedFonts` su `true` e compila il campo di testo con un blocco di testo.

Il secondo blocco di codice definisce un formato di testo con il carattere Times New Roman, imposta le dimensioni del carattere su 30 pixel e applica il formato di testo al campo `lorem_txt`.

Il terzo, e ultimo, blocco di codice definisce e assegna un listener del mouse per l'evento `onMouseMove`. Quando il puntatore del mouse si sposta in orizzontale sullo stage, la proprietà `TextField.thickness` viene modificata e impostata su un valore tra -200 e +200, a seconda del valore corrente di `_xmouse`.

6. Salvare le modifiche al file FLA.

7. Selezionare Controllo > Prova filmato per provare il documento Flash.

Quando si sposta il puntatore del mouse nella metà sinistra dello stage, lo spessore del carattere diminuisce; quando si sposta nella metà destra dello stage, lo spessore del carattere aumenta.

Informazioni sulla condivisione dei caratteri

Se si desidera utilizzare un carattere come elemento di una libreria condivisa, è possibile creare un simbolo di carattere nel pannello Libreria e quindi assegnare a tale simbolo i seguenti attributi:

- Una stringa di identificazione
- Un URL a cui verrà inviato il documento contenente il simbolo di carattere.

È quindi possibile creare un collegamento con il carattere e utilizzarlo in un'applicazione Flash, senza memorizzare il carattere nel file FLA.

Informazioni sul rendering dei caratteri e sul testo con antialiasing

Il rendering dei caratteri in Flash consente di controllare la modalità di visualizzazione del testo in un file SWF, ovvero come viene reso (o *disegnato*) in fase di runtime. La tecnologia di rendering avanzato utilizzata in Flash Player 8 è chiamata FlashType. FlashType utilizza una tecnologia di rendering avanzata per migliorare la leggibilità e la chiarezza del testo con dimensioni di carattere piccole e medie, simile all'applicazione di antialiasing avanzato nei campi di testo. Tale tecnologia è trattata dettagliatamente più avanti in questa sezione.

La funzione di antialiasing consente di smussare il testo in modo che i bordi dei caratteri visualizzati sullo schermo appaiano meno irregolari; è particolarmente utile quando si desidera visualizzare testo con caratteri di piccole dimensioni. L'opzione Antialiasing per il testo allinea i contorni del testo lungo i limiti di pixel in modo da rendere il testo più leggibile, ed è particolarmente efficace per il rendering dei caratteri di piccole dimensioni. È possibile applicare l'antialiasing per ogni campo di testo presente nell'applicazione, anziché per singoli caratteri.

Se è installato Flash Player 7 o versione successiva, l'opzione di antialiasing è supportata per il testo di tipo statico, dinamico e di input. Se è installata una versione precedente di Flash Player, la funzione di antialiasing è supportata solo per il testo statico e le relative opzioni sono disponibili solo per Flash Player 8.

Flash Professional 8 e Flash Basic 8 offrono una tecnologia per la rasterizzazione e il rendering dei caratteri migliorata, chiamata FlashType, che consente di utilizzare caratteri con antialiasing. Flash comprende ora cinque diversi metodi di rendering dei caratteri, che sono disponibili solo per la pubblicazione dei file SWF per Flash Player 8. Se si pubblicano file da utilizzare con Flash Player 7 o versioni precedenti, per i campi di testo è possibile utilizzare soltanto la funzione Antialiasing per animazione.

FlashType è una tecnologia di elevata qualità per il rendering di caratteri che può essere utilizzata con lo strumento di creazione Flash 8 o con ActionScript. La tecnologia FlashType consente di effettuare il rendering di alta qualità dei caratteri con dimensioni ridotte, offrendo un maggiore controllo. FlashType può essere applicata a caratteri incorporati per campi di testo statici, dinamici e di input. Queste capacità migliorate consentono la visualizzazione del testo incorporato allo stesso livello di qualità del testo con caratteri dispositivo, e i caratteri hanno lo stesso aspetto su diverse piattaforme.

I metodi di rendering disponibili per Flash Player 8 sono Caratteri dispositivo, Testo bitmap (nessun antialiasing), per animazione, per leggibilità e personalizzato, che consente di definire un valore personalizzato per spessore e precisione. Per ulteriori informazioni su queste opzioni, vedere “[Opzioni di rendering dei caratteri in Flash](#)” a pagina 445.

NOTA

Quando si aprono i file FLA esistenti in Flash 8, il testo non viene automaticamente aggiornato all'opzione per leggibilità; per sfruttare la tecnologia di rendering FlashType è necessario selezionare i singoli campi di testo e modificare manualmente le impostazioni di antialiasing.

Le funzioni antialiasing avanzate e personalizzate supportano quanto segue:

- Testo ruotato e modificato in scala
- Tutti i caratteri (normale, grassetto o corsivo) fino a dimensioni di 255 pt
- Esportazione nella maggior parte dei formati (ad esempio file JPEG o GIF)

Le funzioni antialiasing avanzate e personalizzate non supportano quanto segue:

- Flash Player 7 o versione precedente
- Testo inclinato o riflesso
- Stampa
- Esportazione dei file nel formato PNG

NOTA

Quando il testo viene animato, l'antialiasing avanzato viene disattivato per migliorare l'aspetto del testo in movimento. Al termine dell'animazione, l'antialiasing viene riattivato.

Un file di esempio nel disco rigido illustra l'applicazione e la modifica di testo con antialiasing in un'applicazione. Viene utilizzata la tecnologia di rendering FlashType per la creazione di testo di piccole dimensioni e ben leggibile. L'esempio illustra anche lo scorrimento veloce e fluido dei campi di testo con l'uso della proprietà `cacheAsBitmap`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `flashtype.fla`.

In Windows, accedere a `unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType`.

In Macintosh, accedere a `Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/FlashType`.

Opzioni di rendering dei caratteri in Flash

In Flash 8 sono disponibili cinque diverse opzioni per il rendering dei caratteri. Per scegliere un'opzione, selezionare il campo di testo e aprire la finestra di ispezione Proprietà. Selezionare un'opzione dal menu a comparsa Metodo di rendering caratteri.

Usa caratteri dispositivo Produce un file SWF di dimensioni inferiori ed esegue il rendering utilizzando i caratteri attualmente installati nel computer dell'utente finale.

Testo bitmap (senza antialiasing) Produce bordi del testo nitidi, senza antialiasing, e un file SWF di dimensioni maggiori in quanto i profili del carattere sono inclusi nel file SWF.

Antialiasing per animazione Produce testo con antialiasing con un'animazione fluida. In alcune situazioni, il testo viene inoltre animato più velocemente poiché, durante l'animazione, non vengono applicati l'allineamento e l'antialiasing. Se si utilizzano caratteri di grandi dimensioni con molte lettere oppure caratteri modificati in scala, non si noteranno miglioramenti delle prestazioni. Questa opzione produce un file SWF di dimensioni maggiori in quanto i profili del carattere sono inclusi nel file SWF.

Antialiasing per leggibilità Per questa opzione viene utilizzato il motore di antialiasing avanzato, che offre testo della massima qualità e leggibilità. Questa opzione produce un file SWF delle massime dimensioni in quanto, oltre ai profili del carattere, include speciali informazioni sull'antialiasing.

Antialiasing personalizzato È analoga all' per leggibilità, ma consente di modificare visivamente i parametri di antialiasing per produrre un aspetto specifico. È utile per produrre l'aspetto migliore possibile per i caratteri nuovi o non comuni.

Per un esempio dell'uso di antialiasing con ActionScript, vedere “[Impostazione dell'antialiasing con ActionScript](#)” a pagina 446.

Informazioni sulla modulazione continua del tratto

La tecnologia di rendering di caratteri FlashType sfrutta le proprietà intrinseche dei campi con distanza per offrire la modulazione continua del tratto (CSM, Continuous Stroke Modulation); ad esempio la modulazione continua dello spessore del tratto e della precisione del bordo del testo. CSM utilizza due parametri di rendering per controllare la mappatura di distanze di campi con distanza a campionamento adattivo (ADF, adaptively sampled distance field) su valori di densità di glifo. I valori ottimali per questi parametri sono soggettivi; possono dipendere dalle preferenze dell'utente, dalle condizioni di illuminazione, dalle proprietà di visualizzazione, dal carattere, dai colori di primo piano e di sfondo e dalla dimensione in punti. La funzione che effettua la mappatura di distanze ADF su valori di densità ha un valore di taglio esterno, sotto al quale i valori vengono impostati su zero, e un valore di taglio interno, sopra al quale i valori vengono impostati su un valore di densità massimo, ad esempio 255.

Impostazione dell'antialiasing con ActionScript

Flash 8 offre due tipi di antialiasing: normale e avanzato. L'antialiasing avanzato è disponibile solo in Flash Player 8 e versioni successive e può essere utilizzato solo se il carattere è incorporato nella libreria e se la proprietà `embedFonts` del campo di testo è impostata su `true`. Per Flash Player 8, l'impostazione predefinita per i campi di testo creati con ActionScript è `normal`.

Per impostare i valori per la proprietà `TextField.antiAliasType`, utilizzare i seguenti valori di stringa:

normal Applica al testo l'antialiasing normale. Questa impostazione corrisponde al tipo di antialiasing utilizzato da Flash Player versione 7 e precedenti.

advanced Applica l'antialiasing avanzato, disponibile in Flash Player 8, che consente di eseguire il rendering dei caratteri con una qualità molto alta anche quando sono di piccole dimensioni. Trova il suo utilizzo ottimale nelle applicazioni con molto testo di piccole dimensioni.

SUGGERIMENTO

Macromedia sconsiglia l'utilizzo di antialiasing avanzato per caratteri con dimensioni maggiori di 48 punti.

Per impostare il testo con antialiasing utilizzando ActionScript, vedere il seguente esempio.

Per utilizzare l'antialiasing avanzato:

1. Creare un nuovo documento Flash e salvarlo come **antialiasing.fla**.
2. Creare due clip filmato sullo stage e assegnare loro i nomi di istanza **normal_mc** e **advanced_mc**.
I due clip filmato verranno utilizzati per alternare i due tipi di antialiasing: normale e avanzato.
3. Aprire il pannello Libreria e selezionare Nuovo carattere dal menu a comparsa nell'angolo superiore destro del pannello.
Si apre la finestra di dialogo Proprietà del simbolo carattere, dove è possibile selezionare un carattere da incorporare nel file SWF (compresi uno stile e le dimensioni del carattere). È inoltre possibile assegnare un nome di carattere visualizzato nella libreria del documento e nel menu a discesa Carattere della finestra di ispezione Proprietà (se sullo stage è selezionato un campo di testo).
 - a. Selezionare il carattere Arial dal menu a discesa Carattere.
 - b. Verificare che non siano selezionate le opzioni Grassetto e Corsivo.
 - c. Impostare le dimensioni su 10 pixel.
 - d. Immettere il nome di carattere **Arial-10 (embedded)**.
 - e. Fare clic su OK.
4. Nella libreria, fare clic con il pulsante destro del mouse sul simbolo del carattere e selezionare Concatenamento dal menu di scelta rapida.
Viene visualizzata la finestra di dialogo Proprietà del concatenamento.
5. Selezionare le opzioni Esporta per ActionScript ed Esporta nel primo fotogramma, inserire l'identificatore di concatenamento **Arial-10**, quindi fare clic su OK.
6. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var text_fmt:TextFormat = new TextFormat();
text_fmt.font = "Arial-10";
text_fmt.size = 10;

this.createTextField("my_txt", 10, 20, 20, 320, 240);
my_txt.autoSize = "left";
my_txt.embedFonts = true;
my_txt.selectable = false;
my_txt.setNewTextFormat(text_fmt);
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
```

```

if (src != undefined) {
    my_txt.text = src;
} else {
    my_txt.text = "unable to load text file.";
}
};

lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

normal_mc.onRelease = function() {
    my_txt antiAliasType = "normal";
};

advanced_mc.onRelease = function() {
    my_txt antiAliasType = "advanced";
};

```

Il codice precedente viene suddiviso in quattro aree principali. Il primo blocco di codice crea un nuovo oggetto `TextFormat`, che specifica un carattere e dimensione di carattere da utilizzare in un campo di testo che verrà creato fra breve. Il carattere specificato, `Arial-10`, è l'identificatore di concatenamento per il simbolo di carattere incorporato in un punto precedente.

Il secondo blocco di codice crea un nuovo campo di testo con il nome di istanza `my_txt`. Per incorporare correttamente il carattere, è necessario impostare `embedFonts` su `true` per l'istanza di campo di testo. Il codice imposta inoltre la formattazione del testo per il nuovo campo di testo sull'oggetto `TextFormat` creato in precedenza.

Il terzo blocco di codice definisce un'istanza `LoadVars` che compila il campo di testo sullo stage con i contenuti di un file di testo esterno. Quando il documento è caricato (ma non analizzato), il contenuto del file viene copiato nella proprietà `my_txt.text`, in modo che sia visualizzato sullo stage.

Il quarto e ultimo blocco di codice definisce i gestori di eventi `onRelease` per i clip filmato `normal_mc` e `advanced_mc`. Quando l'utente fa clic su una di queste opzioni, cambia il tipo di antialiasing per il campo di testo sullo stage.

- 7.** Salvare le modifiche al file FLA.
- 8.** Selezionare Controllo > Prova filmato per provare il documento Flash.
- 9.** Fare clic sul clip filmato `advanced_mc` sullo stage.

Facendo clic sul clip filmato si alterna il tipo di antialiasing tra normale (l'impostazione predefinita) e avanzato. Quando si trattano campi di testo con dimensioni di carattere ridotte, l'impostazione dell'antialiasing su avanzato può migliorare significativamente la leggibilità del testo.

SUGGERIMENTO

La modalità di antialiasing avanzato consente di effettuare il rendering di altissima qualità dei caratteri anche a dimensioni ridotte. Trova il suo utilizzo ottimale nelle applicazioni con molto testo di piccole dimensioni. Macromedia sconsiglia l'utilizzo di antialiasing avanzato per caratteri con dimensioni maggiori di 48 punti.

Per informazioni sulla formattazione di testo con antialiasing, vedere “[Utilizzo di un tipo di adattamento alla griglia](#)” a pagina 456 e “[Informazioni sulla formattazione di testo con antialiasing](#)” a pagina 453.

Un file di esempio nel disco rigido illustra l'applicazione e la modifica di testo con antialiasing in un'applicazione. Viene utilizzata la tecnologia di rendering FlashType per la creazione di testo di piccole dimensioni e ben leggibile. L'esempio illustra anche lo scorrimento veloce e fluido dei campi di testo con l'uso della proprietà `cacheAsBitmap`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `flashtype.fla`.

In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType*.

In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/FlashType*.

Impostazione di tabelle per i caratteri

Se si creano caratteri da utilizzare in file SWF per la distribuzione a sviluppatori Flash, può essere necessaria l'impostazione di tabelle per il controllo della visualizzazione dei caratteri sullo stage.

L'antialiasing avanzato utilizza campi con distanza a campionamento adattivo (ADF) per rappresentare i profili che determinano un glifo (un carattere). Flash utilizza due valori possibili:

- Un valore di taglio esterno, sotto al quale le densità vengono impostate su 0.
- Un valore di taglio interno, sopra al quale le densità vengono impostate su un valore di densità massimo, ad esempio 255.

Tra questi due valori di taglio, la funzione di mappatura è una curva lineare compresa tra 0 al taglio esterno e la densità massima al taglio interno.

La regolazione dei valori di taglio interno ed esterno incide sullo spessore del tratto e sulla precisione del bordo. Lo spazio tra questi due parametri è paragonabile a due volte il raggio del filtro dei metodi di antialiasing classici; uno spazio ridotto fornisce un bordo più nitido, mentre uno spazio più ampio fornisce un bordo più filtrato e sfumato. Quando lo spazio è 0, l'immagine della densità risultante è una bitmap a due livelli. Quando lo spazio è molto ampio, l'immagine della densità risultante ha un bordo simile a un tratto ad acquerello.

Di solito, gli utenti preferiscono bordi nitidi a contrasto elevato quando il testo è di piccole dimensioni, e bordi più sfumati per il testo animato o di grandi dimensioni.

Normalmente, il taglio esterno ha un valore negativo, mentre quello interno ha un valore positivo e il loro punto centrale è di solito vicino allo zero. La regolazione di questi parametri per spostare il punto centrale verso l'infinito negativo aumenta lo spessore del tratto, mentre lo spostamento del punto centrale verso l'infinito positivo diminuisce lo spessore del tratto.



Il taglio esterno dovrebbe sempre essere minore o uguale al taglio interno.

Flash Player include impostazioni di antialiasing avanzato per 10 caratteri di base, per i quali vengono tuttavia fornite le impostazioni di antialiasing avanzato solo per i caratteri con dimensioni da 6 a 20. Per questi caratteri, tutte le dimensioni inferiori a 6 utilizzano le impostazioni per 6, mentre tutte le dimensioni superiori a 20 utilizzano le impostazioni per 20. Gli altri caratteri vengono mappati ai dati dei caratteri forniti. Il metodo `setAdvancedAntialiasingTable()` consente l'impostazione di dati antialiasing personalizzati per altri caratteri e dimensioni di caratteri, o per sostituire le impostazioni predefinite per i caratteri forniti. Per ulteriori informazioni sulla creazione di una tabella di antialiasing, vedere il seguente esempio:

Per creare una tabella di antialiasing avanzato per un carattere incorporato:

1. Creare un nuovo documento Flash e salvarlo come `advancedataable.fla`.
2. Selezionare Nuovo carattere dal menu a comparsa del pannello Libreria.
3. Selezionare Arial dal menu a comparsa Carattere, quindi impostare la dimensione del carattere su **32 punti**.
4. Selezionare le opzioni Grassetto e Corsivo.
5. Immettere il nome di carattere **Arial (embedded)** nella casella di testo Nome, quindi fare clic su OK.
6. Fare clic con il pulsante destro del mouse (Windows) o premere il tasto Ctrl (Macintosh) e fare clic sul simbolo di carattere nella libreria, quindi selezionare Concatenamento.

7. Nella finestra di dialogo Proprietà del concatenamento:
 - a. Digitare **Arial-embedded** nella casella di testo Identificatore.
 - b. Selezionare Esporta per ActionScript ed Esporta nel primo fotogramma.
 - c. Fare clic su OK.
8. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```

import flash.text.TextRenderer;
var arialTable:Array = new Array();
arialTable.push({fontSize:16.0, insideCutoff:0.516,
    outsideCutoff:0.416});
arialTable.push({fontSize:32.0, insideCutoff:2.8, outsideCutoff:-2.8});
TextRenderer.setAdvancedAntialiasingTable("Arial", "bolditalic", "dark",
    arialTable);

var my_fmt:TextFormat = new TextFormat();
my_fmt.align = "justify";
my_fmt.font = "Arial-embedded";
my_fmt.size = 32;

this.createTextField("my_txt", 999, 10, 10, Stage.width-20,
    Stage.height-20);
my_txt antiAliasType = "advanced";
my_txt.embedFonts = true;
my_txt.multiline = true;
my_txt.setNewTextFormat(my_fmt);
my_txt.sharpness = 0;
my_txt.thickness = 0;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
    if (src != undefined) {
        my_txt.text = src + "\n\n" + src;
    } else {
        trace("error downloading text file");
    }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

```

Il codice precedente viene suddiviso in quattro sezioni. La prima sezione di codice importa la classe TextRenderer e definire una nuova tabella di antialiasing per due diverse dimensioni del carattere Arial. La seconda sezione di codice definisce un nuovo oggetto TextFormat, utilizzato per applicare la formattazione di testo al campo di testo (creato nella sezione di codice successiva). La sezione di codice successiva crea un nuovo campo di testo con nome di istanza my_txt, abilita l'antialiasing avanzato, applica l'oggetto formato di testo (creato in precedenza) e abilita il testo multiriga e il ritorno a capo automatico. Il blocco di codice finale definisce un oggetto LoadVars utilizzato per caricare il testo da un file esterno e per compilare il campo di testo sullo stage.

9. Selezionare Controllo > Prova filmato per provare il documento Flash.

Al termine del caricamento del testo dal server remoto, Flash visualizza del testo nel campo di testo e si possono osservare le proprietà di tabella di antialiasing avanzato applicate al campo di testo. Il carattere incorporato sullo stage sembra leggermente sfocato a causa dei valori correnti di `insideCutoff` e `outsideCutoff`.

Informazioni sul layout e sulla formattazione del testo

Mediante ActionScript è possibile controllare layout e formattazione del testo. La classe TextFormat offre un alto grado di controllo sulla visualizzazione del testo in fase di runtime, oltre ad altre forme di formattazione quali i fogli di stile (vedere “[Formattazione del testo con gli stili CSS](#)” a pagina 461) e di testo HTML (vedere “[Uso di un testo in formato HTML](#)” a pagina 475).

Mediante ActionScript è inoltre possibile controllare l'adattamento dei caratteri alla griglia quando si utilizza testo con antialiasing in un file SWF. In questo modo è possibile controllare l'aspetto dei caratteri in fase di runtime. Per un esempio di utilizzo di un tipo di adattamento alla griglia nelle applicazioni, vedere “[Utilizzo di un tipo di adattamento alla griglia](#)” a pagina 456.

Per informazioni generali sui campi di testo, vedere “[Informazioni sui campi di testo](#)” a pagina 417. Per informazioni sulla formattazione di testo, vedere “[Informazioni sulla formattazione di testo con antialiasing](#)” a pagina 453. Per ulteriori informazioni sulla classe TextFormat, vedere “[Uso della classe TextFormat](#)” a pagina 458 e TextFormat nella *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni sul layout e sulla formattazione del testo con la classe TextFormat, consultare le seguenti sezioni:

- “[Informazioni sulla formattazione di testo con antialiasing](#)” a pagina 453
- “[Utilizzo di un tipo di adattamento alla griglia](#)” a pagina 456
- “[Uso della classe TextFormat](#)” a pagina 458
- “[Proprietà predefinite dei nuovi campi di testo](#)” a pagina 460

Informazioni sulla formattazione di testo con antialiasing

Flash 8 introduce due nuove proprietà che possono essere utilizzate nella formattazione di campi di testo quando è abilitato l'antialiasing avanzato: `sharpness` e `thickness`. La proprietà `sharpness` (nitidezza) è la quantità di aliasing applicato all'istanza del campo di testo. Con un valore di nitidezza alto i bordi del carattere incorporato hanno un aspetto irregolare e nitido. L'impostazione di `sharpness` su un valore inferiore visualizza caratteri più smussati e sfocati. L'impostazione della proprietà `thickness` (spessore) è simile all'attivazione della formattazione in grassetto per un campo di testo: più il valore è alto, più il grassetto del carattere è marcato.

Il seguente esempio carica un testo di file in modo dinamico e visualizza del testo sullo stage. Se si sposta il puntatore del mouse lungo l'asse *x* si imposta la nitidezza tra -400 e 400, se si sposta il puntatore del mouse lungo l'asse *y* si imposta lo spessore tra -200 e 200.

Per modificare nitidezza e spessore di un campo di testo:

1. Creare un nuovo documento Flash e salvarlo come `sharpness.fla`.
2. Selezionare Nuovo carattere dal menu a comparsa nell'angolo superiore destro del pannello Libreria.
3. Selezionare Arial dal menu a comparsa Carattere, quindi impostare la dimensione del carattere su 24 punti.
4. Immettere il nome di carattere **Arial-24 (embedded)** nella casella di testo Nome, quindi fare clic su OK.
5. Fare clic con il tasto destro del mouse sul simbolo del carattere nella libreria e selezionare Concatenamento per aprire la finestra di dialogo Proprietà del concatenamento.
6. Impostare l'identificatore del concatenamento su Arial-24, selezionare le caselle di controllo Esporta per ActionScript ed Esporta nel primo fotogramma, quindi fare clic su OK.

7. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 24;
my_fmt.font = "Arial-24";

this.createTextField("lorem_txt", 10, 0, 20, Stage.width, (Stage.height
- 20));
lorem_txt.setNewTextFormat(my_fmt);
lorem_txt.text = "loading...";
lorem_txt.wordWrap = true;
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt antiAliasType = "advanced";

this.createTextField("debug_txt", 100, 0, 0, Stage.width, 20);
debug_txt.autoSize = "left";
debug_txt.background = 0xFFFFFFFF;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
    lorem_txt.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
    lorem_txt.sharpness = (_xmouse * (800 / Stage.width)) - 400;
    lorem_txt.thickness = (_ymouse * (400 / Stage.height)) - 200;
    debug_txt.text = "sharpness=" + Math.round(lorem_txt.sharpness) +
        ", thickness=" + Math.round(lorem_txt.thickness);
};
Mouse.addListener(mouseListener);

};
```

Il codice ActionScript può essere separato in cinque sezioni principali. La prima sezione di codice definisce una nuova istanza di TextFormat che verrà applicata a un campo di testo creato in modo dinamico. Le due sezioni di testo successive creano due nuovi campi di testo sullo stage. Il primo campo di testo, `lorem_txt`, applica l'oggetto di formattazione di testo personalizzata creato in precedenza, abilita i caratteri incorporati e imposta la proprietà `antiAliasType` su `true`. Il secondo campo di testo, `debug_txt`, visualizza i valori di nitidezza e spessore correnti per il campo di testo `lorem_txt`. La quarta sezione di codice crea un oggetto `LoadVars`, responsabile del caricamento del file esterno e della compilazione del campo di testo `lorem_txt`. La quinta e ultima sezione di codice definisce un listener del mouse, chiamato ogni qualvolta il mouse viene spostato sullo stage. I valori correnti per `sharpness` e `thickness` sono calcolati in base alla posizione corrente del puntatore del mouse sullo stage. Le proprietà `sharpness` e `thickness` sono impostate per il campo di testo `lorem_txt` e i valori correnti sono visualizzati nel campo di testo `debug_txt`.

8. Selezionare Controllo > Prova filmato per provare il documento.

Spostare il puntatore del mouse lungo l'asse *x* per modificare la nitidezza del campo di testo. Spostare il puntatore del mouse da sinistra verso destra per aumentare la nitidezza e dare al testo un aspetto più irregolare. Spostare il puntatore del mouse lungo l'asse *y* per modificare lo spessore del campo di testo.

Per ulteriori informazioni sull'utilizzo di testo con antialiasing in un file SWF, vedere “[Impostazione dell'antialiasing con ActionScript](#)” a pagina 446, “[Opzioni di rendering dei caratteri in Flash](#)” a pagina 445 e “[Utilizzo di un tipo di adattamento alla griglia](#)” a pagina 456.

Un file di esempio nel disco rigido illustra l'applicazione e la modifica di testo con antialiasing in un'applicazione. Viene utilizzata la tecnologia di rendering FlashType per la creazione di testo di piccole dimensioni e ben leggibile. L'esempio illustra anche lo scorrimento veloce e fluido dei campi di testo con l'uso della proprietà `cacheAsBitmap`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `flashtype.fla`.

In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType*.

In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/FlashType*.

Utilizzo di un tipo di adattamento alla griglia

Quando si utilizza l'antialiasing avanzato su un campo di testo, sono disponibili tre tipi di adattamento alla griglia:

none Indica adattamento alla griglia. Le linee orizzontali e verticali dei glifi non vengono forzate alla griglia di pixel. Si tratta di un'impostazione ottima per le animazioni o i caratteri con dimensioni elevate.

pixel Indica che le linee orizzontali e verticali spesse vengono adattate alla griglia di pixel. Questa impostazione funziona solo per campi di testo allineati a sinistra. Di solito, fornisce il livello migliore di leggibilità per il testo allineato a sinistra.

subpixel Indica che le linee orizzontali e verticali spesse vengono adattate alla griglia di subpixel su un monitor LCD. Si tratta di un'impostazione ottima per il testo dinamico allineato a destra o al centro, e spesso rappresenta un utile compromesso in termini di qualità tra l'animazione e il testo.

L'esempio che segue illustra come impostare un tipo di adattamento alla griglia su un campo di testo mediante ActionScript.

Per impostare un tipo di adattamento alla griglia su un campo di testo:

1. Creare un nuovo documento Flash e salvarlo come **gridfittype.fla**.
2. Selezionare Nuovo carattere dal menu a comparsa nell'angolo superiore destro del pannello Libreria.
3. Selezionare il carattere Arial dal menu a comparsa Carattere, quindi impostare la dimensione del carattere su 10 punti.
4. Immettere il nome di carattere **Arial-10 (embedded)** nella casella di testo Nome, quindi fare clic su OK.
5. Fare clic con il tasto destro del mouse sul simbolo del carattere nella libreria e selezionare Concatenamento per aprire la finestra di dialogo Proprietà del concatenamento.
6. Impostare l'identificatore del concatenamento su Arial-10, quindi selezionare le caselle di controllo Esporta per ActionScript ed Esporta nel primo fotogramma.
7. Fare clic su OK.

8. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 10;
my_fmt.font = "Arial-10";
var h:Number = Math.floor(Stage.height / 3);

this.createTextField("none_txt", 10, 0, 0, Stage.width, h);
none_txt.antiAliasType = "advanced";
none_txt.embedFonts = true;
none_txt.gridFitType = "none";
none_txt.multiline = true;
none_txt.setNewTextFormat(my_fmt);
none_txt.text = "loading...";
none_txt.wordWrap = true;

this.createTextField("pixel_txt", 20, 0, h, Stage.width, h);
pixel_txt.antiAliasType = "advanced";
pixel_txt.embedFonts = true;
pixel_txt.gridFitType = "pixel";
pixel_txt.multiline = true;
pixel_txt.selectable = false;
pixel_txt.setNewTextFormat(my_fmt);
pixel_txt.text = "loading...";
pixel_txt.wordWrap = true;

this.createTextField("subpixel_txt", 30, 0, h*2, Stage.width, h);
subpixel_txt.antiAliasType = "advanced";
subpixel_txt.embedFonts = true;
subpixel_txt.gridFitType = "subpixel";
subpixel_txt.multiline = true;
subpixel_txt.setNewTextFormat(my_fmt);
subpixel_txt.text = "loading...";
subpixel_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
    if (src != undefined) {
        none_txt.text = "[antiAliasType=none]\n" + src;
        pixel_txt.text = "[antiAliasType=pixel]\n" + src;
        subpixel_txt.text = "[antiAliasType=subpixel]\n" + src;
    } else {
        trace("unable to load text file");
    }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");


```

Il codice ActionScript può essere separato in cinque sezioni. La prima sezione definisce un nuovo oggetto formato di testo che specifica due proprietà, `size` e `font`. La proprietà `font` fa riferimento all'identificatore di concatenamento del simbolo di carattere che si trova nella libreria del documento. La seconda, la terza e la quarta sezione del codice creano tre nuovi campi di testo dinamico sullo stage e impostano alcune proprietà comuni: `antiAliasType`, che deve essere impostata su `advanced`, `embedFonts`, impostata su `true`, `multiline` e `wordWrap`. Ciascuna sezione applica inoltre l'oggetto formato di testo creato in una sezione precedente e impone il tipo di adattamento alla griglia su `normal`, `pixel` o `subpixel`. La quinta e ultima sezione crea un'istanza `LoadVars`, che carica il contenuto di un file di testo esterno in ciascun campo di testo creato con il codice.

9. Salvare il documento e selezionare Controllo > Prova filmato per provare il file SWF.

Ciascun campo di testo deve essere inizializzato con il valore "loading...". Dopo il caricamento del file esterno, ogni campo di testo visualizza del testo di esempio formattato con un diverso tipo di adattamento alla griglia.

SUGGERIMENTO

La tecnologia di rendering FlashType utilizza l'adattamento alla griglia solo con una rotazione di 0°.

Uso della classe TextFormat

È possibile utilizzare la classe `TextFormat` per impostare le proprietà di formattazione di un campo di testo. La classe `TextFormat` incorpora le informazioni relative alla formattazione dei caratteri e dei paragrafi. Le informazioni sulla formattazione dei caratteri descrivono l'aspetto dei singoli caratteri: nome del carattere, dimensione in punti, colore e URL associato. Le informazioni sulla formattazione dei paragrafi consentono di descrivere l'aspetto di un paragrafo: margine sinistro, margine destro, rientro della prima riga e allineamento a sinistra, a destra o al centro.

Per utilizzare la classe `TextFormat`, creare prima un oggetto `TextFormat` e impostarne i relativi stili di formattazione di paragrafi e caratteri, quindi applicare l'oggetto `TextFormat` a un campo di testo utilizzando i metodi `TextField.setTextFormat()` o `TextField.setNewTextFormat()`.

Il metodo `setTextFormat()` modifica il formato del testo applicato a singoli caratteri, a gruppi di caratteri o all'intero corpo del testo presente in un campo di testo. Tuttavia, il nuovo testo immesso, ovvero il testo immesso da un utente o mediante ActionScript, non assume la formattazione specificata da una chiamata `setTextFormat()`. Per specificare la formattazione predefinita per il nuovo testo inserito, utilizzare `TextField.setNewTextFormat()`. Per ulteriori informazioni, vedere `setTextFormat (TextField.setTextFormat method)` e `setNewTextFormat (TextField.setNewTextFormat method)` nella *Guida di riferimento di ActionScript 2.0*.

Per formattare un campo di testo con la classe `TextFormat`:

1. In un nuovo documento Flash, creare un campo di testo sullo stage utilizzando lo strumento Testo.
Digitare un testo nel campo di testo sullo stage, ad esempio **Grassetto corsivo 24 punti**.
2. Nella finestra di ispezione Proprietà, digitare **myText_txt** nella casella di testo Nome istanza, selezionare Dinamico dal menu a comparsa Tipo testo e quindi selezionare Multiriga dal menu a comparsa Tipo linea.
3. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
4. Immettere il seguente codice nel pannello Azioni per creare un oggetto `TextFormat`, quindi impostare le proprietà `bold` e `italic` su `true` e la proprietà `size` su `24`:

```
// Crea un oggetto TextFormat.  
var txt_fmt:TextFormat = new TextFormat();  
// Specifica la formattazione del carattere e del paragrafo.  
txt_fmt.bold = true;  
txt_fmt.italic = true;  
txt_fmt.size = 24;
```

5. Applicare l'oggetto `TextFormat` al campo di testo creato al punto 1 utilizzando `TextField.setTextFormat()`:

```
myText_txt.setTextFormat(txt_fmt);
```

Questa versione di `setTextFormat()` consente di applicare la formattazione specificata a tutto il campo di testo. Esistono altre due versioni di questo metodo che consentono di applicare la formattazione a caratteri singoli o a gruppi di caratteri. Il codice seguente, ad esempio, consente di applicare il grassetto, il corsivo e la dimensione di 24 punti ai tre caratteri immessi nel campo di testo:

```
myText_txt.setTextFormat(0, 3, txt_fmt);
```

Per ulteriori informazioni, vedere `setTextFormat (TextField.setTextFormat method)` nella *Guida di riferimento di ActionScript 2.0*.

6. Selezionare Controllo > Prova filmato per provare l'applicazione.

Per ulteriori informazioni sull'uso della classe TextFormat, consultare i seguenti argomenti:

- “Proprietà predefinite dei nuovi campi di testo” a pagina 460
- “Formattazione del testo con gli stili CSS” a pagina 461

Proprietà predefinite dei nuovi campi di testo

I campi di testo creati in fase di runtime con `createTextField()` ricevono un oggetto TextFormat predefinito con le seguenti proprietà:

```
align = "left"
blockIndent = 0
bold = false
bullet = false
color = 0x000000
font = "Times New Roman" (default font is Times on Mac OS X)
indent = 0
italic = false
kerning = false
leading = 0
leftMargin = 0
letterSpacing = 0
rightMargin = 0
size = 12
tabStops = [] (empty array)
target = ""
underline = false
url = ""
```



La proprietà predefinita per il carattere in Mac OS X è Times.

Per un elenco completo dei metodi di TextFormat con le relative descrizioni, vedere TextFormat nella *Guida di riferimento di ActionScript 2.0*.

Formattazione del testo con gli stili CSS

I fogli di stile CSS (Cascading Style Sheets) consentono di lavorare con stili di testo che possono essere applicati a documenti HTML o XML. Un foglio di stile è un insieme di regole di formattazione che specificano come formattare gli elementi HTML o XML. Ogni regola associa un nome di stile, o *selettori*, a una o più proprietà di stile e ai rispettivi valori. Il seguente stile definisce ad esempio un selettori denominato `bodyText`:

```
.bodyText {  
    text-align: left  
}
```

È possibile creare stili che ridefiniscono i tag di formattazione HTML incorporati utilizzati da Flash Player, ad esempio `<p>` e ``. È anche possibile creare *classi* di stile che possono essere applicate ad elementi HTML specifici utilizzando l'attributo `class` dei tag `<p>` o ``, oppure definire nuovi tag.

È possibile utilizzare la classe `TextField.StyleSheet` per utilizzare i fogli di stile del testo. Sebbene la classe `TextField` possa essere utilizzata con Flash Player 6, per la classe `TextField.StyleSheet` è necessario che i file SWF siano eseguiti in Flash Player 7 o versione successiva. È possibile caricare gli stili da un file CSS esterno oppure crearli all'origine utilizzando ActionScript. Per applicare un foglio di stile a un campo di testo che contiene testo formattato in HTML o XML, utilizzare la proprietà `TextField.styleSheet`. Gli stili definiti nel foglio di stile vengono mappati automaticamente ai tag definiti nel documento HTML o XML.

Per utilizzare i fogli di stile è necessario effettuare le tre operazioni di base seguenti:

- Creare un oggetto foglio di stile dalla classe `TextField.StyleSheet`. Per ulteriori informazioni, vedere `StyleSheet` (`TextField.StyleSheet`) nella *Guida di riferimento di ActionScript 2.0*.
- Aggiungere gli stili all'oggetto foglio di stile, caricandoli da un file CSS esterno o creandoli con ActionScript.
- Assegnare il foglio di stile a un oggetto `TextField` che contiene testo in formato HTML o XML.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Proprietà CSS supportate](#)” a pagina 462
- “[Creazione di un oggetto foglio di stile](#)” a pagina 463
- “[Caricamento di file CSS esterni](#)” a pagina 464
- “[Creazione di nuovi stili con ActionScript](#)” a pagina 466
- “[Applicazione di stili all'oggetto `TextField`](#)” a pagina 466

- “Come applicare un foglio di stile a un componente TextArea” a pagina 467
- “Combinazione di stili” a pagina 468
- “Uso delle classi di stile” a pagina 468
- “Assegnazione dello stile a tag HTML incorporati” a pagina 469
- “Esempio di utilizzo di stili con HTML” a pagina 470
- “Uso degli stili per la definizione di nuovi tag” a pagina 472
- “Un esempio di utilizzo di stili con XML” a pagina 473

È possibile trovare un file sorgente di esempio, *formattedText.fla*, nella cartella Samples sul disco rigido, che illustra come applicare la formattazione CSS a testo caricato in un file SWF in fase di runtime.

In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText*.

Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/LoadText*.

Proprietà CSS supportate

Flash Player supporta un sottoinsieme di proprietà della specifica CSS1 originale (www.w3.org/TR/REC-CSS1). Nella tabella seguente sono riportate le proprietà e i valori CSS supportati e i nomi di proprietà di ActionScript corrispondenti. Ogni nome di proprietà di ActionScript deriva dal nome della relativa proprietà CSS; il trattino viene omesso e il carattere che segue è maiuscolo.

Proprietà CSS	Proprietà ActionScript	Uso e valori supportati
text-align	textAlign	I valori riconosciuti sono <code>left</code> , <code>center</code> , <code>right</code> e <code>justify</code> .
font-size	fontSize	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
text-decoration	textDecoration	I valori riconosciuti sono <code>none</code> e <code>underline</code> .
margin-left	marginLeft	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
margin-right	marginRight	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.

Proprietà CSS	Proprietà ActionScript	Uso e valori supportati
font-weight	fontWeight	I valori riconosciuti sono <code>normal</code> e <code>bold</code> .
crenatura	crenatura	I valori riconosciuti sono <code>true</code> e <code>false</code> .
font-style	fontStyle	I valori riconosciuti sono <code>normal</code> e <code>italic</code> .
letterSpacing	letterSpacing	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
text-indent	textIndent	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
font-family	fontFamily	Elenco di caratteri utilizzabili separati da una virgola, in ordine discendente di preferenza. È possibile utilizzare qualsiasi nome della famiglia di caratteri. Se si specifica un nome di carattere generico, questo viene convertito in un carattere di dispositivo adeguato. Sono disponibili le seguenti conversioni di caratteri: <code>mono</code> viene convertito in <code>_typewriter</code> , <code>sans-serif</code> viene convertito in <code>_sans</code> e <code>serif</code> diviene <code>_serif</code> .
color	color	Sono supportati solo i valori colore esadecimali. Non sono supportati i nomi dei colori, ad esempio <code>blue</code> . I colori devono essere scritti nel seguente formato: <code>#FF0000</code> .

Creazione di un oggetto foglio di stile

I fogli di stile CSS sono rappresentati in ActionScript dalla classe `TextField.StyleSheet`, disponibile solo per i file SWF creati per Flash Player 7 o versioni successive. Per creare un oggetto del foglio di stile, chiamare la funzione di costruzione della classe `TextField.StyleSheet`:

```
var newStyle:TextField.StyleSheet = new TextField.StyleSheet();
```

Per aggiungere stili all'oggetto foglio di stile, è possibile caricare un file CSS esterno nell'oggetto oppure definire gli stili in ActionScript. Vedere “[Caricamento di file CSS esterni a pagina 464](#)” e “[Creazione di nuovi stili con ActionScript a pagina 466](#)”.

È possibile trovare un file sorgente di esempio, `formattedText.fla`, nella cartella Samples sul disco rigido, che illustra come applicare la formattazione CSS a testo caricato in un file SWF in fase di runtime.

In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText*.

Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/LoadText*.

Caricamento di file CSS esterni

È possibile definire gli stili in un file CSS esterno e quindi caricare questo file nell'oggetto di un foglio di stile. Gli stili definiti nel file CSS vengono aggiunti nell'oggetto del foglio di stile. Per caricare un file CSS esterno, utilizzare il metodo `load()` della classe `TextField.StyleSheet`. Per stabilire quando il file CSS è stato caricato completamente, utilizzare il gestore di eventi `onLoad` dell'oggetto foglio di stile.

Nell'esempio seguente, viene creato e caricato un file CSS esterno, utilizzando il metodo `TextField.StyleSheet.getStyleNames()` per recuperare i nomi degli stili caricati.

Per caricare un foglio di stile:

1. Creare un nuovo file utilizzando l'editor di testo o l'editor CSS desiderato.
2. Aggiungere al file le seguenti definizioni di stile:

```
.bodyText {  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 12px;  
}  
  
.headline {  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 24px;  
}
```

3. Salvare il file CSS con il nome `styles.css`.
4. In Flash, creare un nuovo file FLA.
5. Nella finestra della Linea temporale (Finestra > Linea temporale), selezionare Livello 1.
6. Aprire il pannello Azioni (Finestra > Azioni).

7. Aggiungere il codice seguente nel pannello Azioni:

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.onLoad = function(success:Boolean):Void {
    if (success) {
        // visualizza i nomi stile.
        trace(this.getStyleNames());
    } else {
        trace("Error loading CSS file.");
    }
};
styles.load("styles.css");
```



Nel frammento di codice riportato sopra, `this.getStyleNames()` fa riferimento all'oggetto `styles` costruito nella prima riga del codice ActionScript.

8. Salvare il file FLA nella stessa directory che contiene il file `styles.css`.

9. Provare il documento Flash selezionando Controllo > Prova filmato.

I nomi dei due stili dovrebbero apparire nel pannello Output:

`.bodyText,.headline`

Se nel pannello Output viene visualizzato un messaggio analogo a "Errore di caricamento del file CSS", verificare che il file FLA e il file CSS si trovino nella stessa directory e che il nome del file CSS sia stato digitato correttamente.

Come per tutti gli altri metodi ActionScript che carichano i dati in rete, il file CSS deve trovarsi nello stesso dominio del file SWF che carica il file. Vedere [“Accesso a più domini e sottodomini tra file SWF”](#) a pagina 754. Per ulteriori informazioni sull'uso dei fogli di stile CSS con Flash, vedere `StyleSheet` (`TextField.StyleSheet`) nella *Guida di riferimento di ActionScript 2.0*.

È possibile trovare un file sorgente di esempio, `formattedText.fla`, nella cartella Samples sul disco rigido, che illustra come applicare la formattazione CSS a testo caricato in un file SWF in fase di runtime.

In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText*.

Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/LoadText*.

Creazione di nuovi stili con ActionScript

È possibile creare i nuovi stili di testo con ActionScript utilizzando il metodo `setStyle()` della classe `TextField.StyleSheet`. Tale metodo accetta due parametri: il nome dello stile e un oggetto che definisce le proprietà dello stile.

Il seguente codice crea ad esempio un oggetto foglio di stile denominato `styles` che definisce due stili identici a quelli importati in precedenza (vedere “[Caricamento di file CSS esterni a pagina 464](#)”):

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("bodyText",
  {fontFamily: 'Arial,Helvetica,sans-serif',
   fontSize: '12px'}
);
styles.setStyle("headline",
  {fontFamily: 'Arial,Helvetica,sans-serif',
   fontSize: '24px'}
);
```

Applicazione di stili all'oggetto `TextField`

Per applicare un oggetto foglio di stile a un oggetto `TextField`, assegnare l'oggetto foglio di stile alla proprietà `styleSheet` del campo di testo.

```
textObj_txt.styleSheet = styles;
```



La proprietà `TextField.styleSheet` non deve essere confusa con la classe `TextField.StyleSheet`. L'uso della lettera maiuscola indica la differenza.

Quando si assegna un oggetto foglio di stile a un oggetto `TextField`, si verificano le seguenti modifiche nel normale comportamento di un campo di testo:

- Le proprietà `text` e `htmlText` del campo di testo e qualsiasi variabile associata al campo di testo contengono sempre lo stesso valore e presentano un comportamento identico.
- Il campo di testo diviene di sola lettura e non può più essere modificato.
- I metodi `setTextFormat()` e `replaceSel()` della classe `TextField` non funzionano più in associazione al campo di testo. Il solo modo per modificare il campo consiste nel cambiare le proprietà `text` o `htmlText` del campo di testo oppure nel modificare la variabile associata al campo di testo.
- Qualsiasi testo assegnato alla proprietà `text`, alla proprietà `htmlText` o alla variabile associata del campo di testo, viene memorizzato testualmente; qualsiasi informazione scritta su una di queste proprietà può essere recuperata nella forma originale del testo.

Come applicare un foglio di stile a un componente TextArea

Per applicare un foglio di stile a un componente TextArea, creare un oggetto foglio di stile e assegnare all'oggetto stili HTML con l'ausilio della classe `TextField.StyleSheet`. In secondo luogo, assegnare il foglio di stile alla proprietà `styleSheet` del componente `TextArea`.

Gli esempi seguenti creano l'oggetto foglio di stile `styles` e lo assegnano all'istanza del componente `myTextArea`.

Uso di un foglio di stile con un componente `TextArea`:

1. Creare un nuovo documento Flash e salvarlo come `textareastyle.fla`.
2. Trascinare un componente `TextArea` nello stage dalla cartella User Interface del pannello Componenti e assegnarvi il nome di istanza `myTextArea`.
3. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
// Create a new style sheet object and set styles for it.  
var styles:TextField.StyleSheet = new TextField.StyleSheet();  
styles.setStyle("html", {fontFamily:'Arial,Helvetica,sans-serif',  
    fontSize:'12px',  
    color:'#0000FF'});  
styles.setStyle("body", {color:'#00CCFF',  
    textDecoration:'underline'});  
styles.setStyle("h1", {fontFamily:'Arial,Helvetica,sans-serif',  
    fontSize:'24px',  
    color:'#006600'});  
  
/* Assign the style sheet object to myTextArea component. Set html  
   property to true, set styleSheet property to the style sheet object.  
 */  
myTextArea.styleSheet = styles;  
myTextArea.html = true;  
  
var myVars:LoadVars = new LoadVars();  
// Define onData handler and load text to be displayed.  
myVars.onData = function(myStr:String):Void {  
    if (myStr != undefined) {  
        myTextArea.text = myStr;  
    } else {  
        trace("Unable to load text file.");  
    }  
};  
myVars.load("http://www.helpexamples.com/flash/myText.htm");
```

Il codice crea una nuova istanza `TextField.StyleSheet` che definisce tre stili, per i tag HTML `html`, `body` e `h1`. Quindi l'oggetto foglio di stile viene applicato al componente `TextArea` e viene abilitata la formattazione HTML. Il codice ActionScript restante definisce un oggetto `LoadVars` che carica un file HTML esterno e compila l'area di testo con il testo caricato.

4. Selezionare Controllo > Prova filmato per provare il documento Flash.

Combinazione di stili

Gli stili CSS in Flash Player possono essere combinati. In altre parole, quando vengono nidificati, ogni livello di nidificazione può fornire informazioni di stile che contribuiscono al risultato di formattazione finale.

L'esempio seguente illustra alcuni dati XML assegnati a un campo di testo:

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text, with one
<emphasized>emphatic</emphasized> word.</mainBody>
```

Per la parola *emphatic* nel testo riportato sopra, lo stile `emphasized` è nidificato all'interno dello stile `mainBody`. Lo stile `mainBody` incide sulle regole del colore, della dimensione caratteri e della decorazione. Lo stile `emphasized` aggiunge una regola di spessore caratteri a tali regole. La parola *emphatic* sarà formattata utilizzando una combinazione delle regole specificate da `mainBody` e `emphasized`.

Uso delle classi di stile

È possibile creare "classi" di stile (non vere e proprie classi ActionScript 2.0) che possono essere applicate ai tag `<p>` o `` utilizzando uno degli attributi `class` del tag. Quando applicato a un tag `<p>`, lo stile influisce sull'intero paragrafo. Inoltre, utilizzando il tag ``, è possibile applicare uno stile a una porzione di testo che usa una classe di stile.

Ad esempio, il seguente foglio di stile definisce due classi di stile: `mainBody` e `emphasis`:

```
.mainBody {
    font-family: Arial,Helvetica,sans-serif;
    font-size: 24px;
}
.emphasis {
    color: #666666;
    font-style: italic;
}
```

All'interno del testo HTML assegnato a un campo di testo è possibile applicare questi stili ai tag `<p>` e ``, come illustrato nel segmento di codice seguente:

```
<p class='mainBody'>This is <span class='emphasis'>really exciting!</span></p>
```

Assegnazione dello stile a tag HTML incorporati

Flash Player supporta un sottoinsieme di tag HTML. Per ulteriori informazioni, consultare “[Uso di un testo in formato HTML](#)” a pagina 475. È possibile assegnare uno stile CSS a ogni istanza di un tag HTML incorporato che viene visualizzato nel campo di testo. Il codice seguente, ad esempio, definisce uno stile per il tag HTML `<p>` incorporato. A tutte le istanze del tag viene assegnato uno stile in base alla regola di stile.

```
p {  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 12px;  
    display: inline;  
}
```

La tabella seguente indica a quali tag HTML incorporati è possibile applicare uno stile e la modalità di assegnazione dello stile.

Nome stile	Modalità di applicazione dello stile
<code>p</code>	Influisce su tutti i tag <code><p></code> .
<code>body</code>	Influisce su tutti i tag <code><body></code> . Lo stile <code>p</code> , se specificato, ha la priorità rispetto allo stile <code>body</code> .
<code>li</code>	Influisce su tutti i tag di punto elenco <code></code> .
<code>a</code>	Influisce su tutti i tag di ancoraggio <code><a></code> .
<code>a:link</code>	Influisce su tutti i tag di ancoraggio <code><a></code> . Questo stile viene applicato dopo ogni stile <code>a</code> .
<code>a:hover</code>	Applicato a un tag di ancoraggio <code><a></code> quando il mouse viene spostato sul collegamento. Questo stile viene applicato dopo ogni stile <code>a</code> e <code>a:link</code> . Quando il mouse viene spostato fuori dal collegamento, lo stile <code>a:hover</code> viene eliminato dal collegamento.
<code>a:active</code>	Applicato a un tag <code><a></code> di ancoraggio facendo clic con il mouse sopra il collegamento. Questo stile viene applicato dopo ogni stile <code>a</code> e <code>a:link</code> . Quando viene rilasciato il pulsante del mouse, lo stile <code>a:active</code> viene eliminato dal collegamento.

Esempio di utilizzo di stili con HTML

Questa sezione analizza un esempio di utilizzo di stili con tag HTML. Viene creato un foglio di stile che assegna uno stile a tag incorporati e definisce classi di stile. Questo foglio di stile viene quindi applicato a un oggetto TextField che contiene testo in formato HTML.

Per formattare testo HTML con un foglio di stile:

1. Creare un nuovo file nell'editor CSS o di testo desiderato.
2. Aggiungere al file le seguenti definizioni del foglio di stile:

```
p {  
    color: #000000;  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 12px;  
    display: inline;  
}  
  
a:link {  
    color: #FF0000;  
}  
  
a:hover{  
    text-decoration: underline;  
}  
  
.headline {  
    color: #000000;  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 18px;  
    font-weight: bold;  
    display: block;  
}  
  
.byline {  
    color: #666600;  
    font-style: italic;  
    font-weight: bold;  
    display: inline;  
}
```

Questo foglio di stile definisce stili per due tag HTML (<p> e <a>) incorporati che saranno applicati a tutte le istanze dei tag. Inoltre, definisce due classi di stile (.headline e .byline) che saranno applicate a paragrafi e a estensioni di testo specifici.

3. Salvare il file come **html_styles.css**.

- 4.** Creare un nuovo file di testo in un editor HTML o di testo e salvare il documento come **myText.htm**.

Aggiungere al file il testo seguente:

```
<p class='headline'>Flash adds FlashType rendering technology!</p><span class='byline'>San Francisco, CA</span>--Macromedia Inc.  
announced today a new version of Flash that features a brand new font  
rendering technology called FlashType, most excellent at rendering  
small text with incredible clarity and consistency across platforms.  
For more information, visit the <a href='http://  
www.macromedia.com'>Macromedia Flash web site.</a></p>
```



Se si copia e si incolla questa stringa di testo, fare attenzione a rimuovere eventuali interruzioni di riga aggiunte nella stringa di testo.

- 5.** Creare un nuovo documento Flash nello strumento di creazione Flash.
- 6.** Selezionare il primo fotogramma nel Livello 1 nella Linea temporale (Finestra > Linea temporale).
- 7.** Aprire il pannello Azioni (Finestra > Azioni) e immettere il seguente codice nel pannello:

```
this.createTextField("news_txt", 99, 50, 50, 450, 300);  
news_txt.border = true;  
news_txt.html = true;  
news_txt.multiline = true;  
news_txt.wordWrap = true;  
// Create a new style sheet and LoadVars object.  
var myVars_lv:LoadVars = new LoadVars();  
var styles:TextField.StyleSheet = new TextField.StyleSheet();  
// Location of CSS and text files to load.  
var txt_url:String = "myText.htm";  
var css_url:String = "html_styles.css";  
// Define onData handler and load text to display.  
myVars_lv.onData = function(src:String):Void {  
    if (src != undefined) {  
        news_txt.htmlText = src;  
    } else {  
        trace("Unable to load HTML file");  
    }  
};  
myVars_lv.load(txt_url);  
  
// Define onLoad handler and Load CSS file.  
styles.onLoad = function(success:Boolean):Void {  
    if (success) {  
        /* Se il foglio di stile caricato non presenta errori,  
           assegnarlo all'oggetto testo,  
           quindi assegnare il testo HTML al campo di testo. */  
        news_txt.styleSheet = styles;
```

```

        news_txt.text = storyText;
    } else {
        trace("Unable to load CSS file.");
    }
};

styles.load(css_url);

```

NOTA

Nel codice ActionScript riportato come esempio, il testo viene caricato da un file esterno. Per ulteriori informazioni sul caricamento di dati esterni, consultare il [Capitolo 15, “Operazioni con immagini, audio e video”](#)

8. Salvare il file come **news_html.fla** nella stessa directory che contiene il file CSS creato al punto 3.
9. Selezionare Controllo > Prova filmato per visualizzare gli stili applicati al testo HTML automaticamente.

Uso degli stili per la definizione di nuovi tag

Se in un foglio di stile viene definito un nuovo stile, questo può essere utilizzato come un tag, analogamente a un tag HTML incorporato. Ad esempio, se un foglio di stile definisce uno stile CSS denominato `sectionHeading`, è possibile utilizzare `<sectionHeading>` come elemento in qualsiasi campo di testo associato al foglio di stile. In questo modo, se si desidera, è possibile assegnare testo in formato XML direttamente a un campo di testo, affinché il testo venga formattato automaticamente in base alle regole presenti nel foglio di stile.

Il seguente foglio di stile crea ad esempio i nuovi stili `sectionHeading`, `mainBody` e `emphasized`:

```

.sectionHeading {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 18px;
    display: block
}
.mainBody {
    color: #000099;
    text-decoration: underline;
    font-size: 12px;
    display: block
}
.emphasized {
    font-weight: bold;
    display: inline
}

```

È possibile quindi immettere in un campo di testo associato al foglio di stile il seguente testo in formato XML:

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text,
with one <emphasized>emphatic</emphasized> word.
</mainBody>
```

Un esempio di utilizzo di stili con XML

In questa sezione viene creato un file FLA con testo in formato XML. Il foglio di stile viene creato con l'ausilio di codice ActionScript e non mediante l'importazione degli stili da un file CSS, come già illustrato in “[Esempio di utilizzo di stili con HTML](#)” a pagina 470

Per formattare testo XML con un foglio di stile:

1. In Flash, creare un file FLA.
2. Utilizzando lo strumento Testo, creare un campo di testo con larghezza e altezza rispettivamente di 400 e 300 pixel circa.
3. Aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà) e selezionare il campo di testo.
4. Nella finestra di ispezione Proprietà, selezionare Testo dinamico dal menu Tipo testo, selezionare Multiriga dal menu Tipo linea, quindi selezionare l'opzione Rendi il testo come HTML e digitare news_txt nella casella di testo Nome istanza.
5. Sul Livello 1 della Linea temporale (Finestra > Linea temporale) selezionare il primo fotogramma.
6. Per creare un oggetto foglio di stile, aprire il pannello Azioni (Finestra > Azioni) e aggiungere il seguente codice al pannello Azioni:

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("mainBody", {
    color:'#000000',
    fontFamily:'Arial,Helvetica,sans-serif',
    fontSize:'12',
    display:'block'
});
styles.setStyle("title", {
    color:'#000000',
    fontFamily:'Arial,Helvetica,sans-serif',
    fontSize:'18',
    display:'block',
    fontWeight:'bold'
});
styles.setStyle("byline", {
    color:'#666600',
```

```

        fontWeight:'bold',
        fontStyle:'italic',
        display:'inline'
    });
    styles.setStyle("a:link", {
        color:'#FF0000'
    });
    styles.setStyle("a:hover", {
        textDecoration:'underline'
    });
}

```

Questo codice crea un nuovo oggetto foglio di stile denominato `styles` che definisce gli stili tramite il metodo `setStyle()`. Gli stili corrispondono esattamente a quelli creati in precedenza in questo capitolo in un file CSS esterno.

- Per creare il testo XML da assegnare al campo di testo, aprire un editor di testo e immettere il testo seguente in un nuovo documento:

```
<story><title>Flash now has FlashType</title><mainBody><byline>San Francisco, CA</byline>--Macromedia Inc. announced today a new version of Flash that features the new FlashType rendering technology. For more information, visit the <a href="http://www.macromedia.com">Macromedia Flash website</a></mainBody></story>
```

NOTA

Se si copia e si incolla questa stringa di testo, fare attenzione a rimuovere eventuali interruzioni di riga aggiunte nella stringa di testo. Nel pannello Azioni, selezionare Caratteri nascosti dal menu a comparsa ed eliminare le interruzioni di riga in eccesso.

- Salvare il file di testo come `story.xml`.
- In Flash, aggiungere il seguente codice nel pannello Azioni, di seguito al codice indicato nel punto 6.

Il codice carica il documento `story.xml`, assegna l'oggetto del foglio di stile alla proprietà `styleSheet` del campo di testo e assegna il testo XML al campo di testo:

```

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean):Void {
    if (success) {
        news_txt.styleSheet = styles;
        news_txt.text = my_xml;
    } else {
        trace("Error loading XML.");
    }
}
my_xml.load("story.xml");

```

NOTA

Nel codice ActionScript riportato come esempio, i dati XML vengono caricati da un file esterno. Per ulteriori informazioni sul caricamento di dati esterni, consultare il [Capitolo 15, “Operazioni con immagini, audio e video”](#)

- 10.** Salvare il file come **news_xml.fla** nella stessa cartella di **story.xml**.
- 11.** Eseguire il filmato (Controllo > Prova filmato) per visualizzare gli stili applicati automaticamente al testo nel campo di testo.

Uso di un testo in formato HTML

Flash Player supporta un sottoinsieme di tag HTML standard, quali `<p>` e `<| i>`, utilizzabili per specificare lo stile del testo in qualsiasi campo di testo di input o dinamico. In Flash Player 7 e nelle versioni successive, i campi di testo supportano anche il tag ``, che consente di incorporare file JPEG, GIF, PNG e SWF, e i clip filmato in un campo di testo. In Flash Player il testo si dispone automaticamente attorno alle immagini incorporate nei campi di testo nello stesso modo in cui in un browser il testo si dispone intorno alle immagini incorporate in una pagina HTML. Per ulteriori informazioni, vedere “[Informazioni sull'incorporamento di immagini, file SWF e clip filmato in campi di testo](#)” a pagina 486.

Flash Player supporta anche il tag `<textformat>`, che consente di applicare gli stili di formattazione paragrafo della classe `TextFormat` a campi di testo abilitati per testo HTML. Per ulteriori informazioni, vedere “[Uso della classe TextFormat](#)” a pagina 458.

Per ulteriori informazioni sul testo in formato HTML, consultare i seguenti argomenti:

- “[Proprietà e sintassi necessarie per l'uso di testo in formato HTML](#)” a pagina 475
- “[Informazioni sui tag HTML supportati](#)” a pagina 477
- “[Informazioni sulle entità HTML](#)” a pagina 485
- “[Informazioni sull'incorporamento di immagini, file SWF e clip filmato in campi di testo](#)” a pagina 486

Proprietà e sintassi necessarie per l'uso di testo in formato HTML

Per inserire codice HTML in un campo di testo, impostare diverse proprietà del campo di testo nella finestra di ispezione Proprietà o con l'ausilio di ActionScript:

- Attivare la formattazione HTML del campo di testo selezionando l'opzione Rendi il testo come HTML nella finestra di ispezione Proprietà oppure impostando la proprietà `html` del campo di testo su `true`.
- Per utilizzare tag HTML come `<p>`, `
` e ``, impostare il campo di testo come multiriga selezionando l'opzione Multiriga nella finestra di ispezione Proprietà o impostando la proprietà `multiline` del campo di testo su `true`.

- In ActionScript, impostare il valore di `TextField.htmlText` sulla stringa in formato HTML che si desidera visualizzare.

Il seguente codice attiva ad esempio la formattazione HTML per un campo di testo denominato `headline_txt`, quindi assegna un testo HTML al campo.

```
this.createTextField("headline_txt", 1, 10, 10, 500, 300);
headline_txt.html = true;
headline_txt.wordWrap = true;
headline_txt.multiline = true;
headline_txt.htmlText = "<font face='Times New Roman' size='25'>This is how
you assign HTML text to a text field.</font><br>It's very useful.<br>";
```

Per eseguire correttamente il rendering dell'HTML, la sintassi deve essere corretta. Gli attributi dei tag HTML devono essere racchiusi tra virgolette semplici ('') o doppie (""). Se i valori degli attributi non vengono racchiusi tra virgolette si potrebbero ottenere risultati imprevisti, ad esempio un rendering scorretto del testo. Il rendering del seguente frammento HTML, ad esempio, *non* viene eseguito correttamente in Flash Player perché il valore assegnato all'attributo `align` (`left`) non è stato racchiuso tra virgolette:

```
this.createTextField("myField_txt", 10, 10, 10, 400, 200);
myField_txt.html = true;
myField_txt.htmlText = "<p align=left>This is left-aligned text</p>";
```

Se i valori degli attributi vengono racchiusi tra virgolette doppie, anteporre il carattere *escape* alle virgolette (\"). Sono accettabili le seguenti operazioni:

```
myField_txt.htmlText = "<p align='left'>This uses single quotes</p>";
myField_txt.htmlText = "<p align=\"left\">This uses escaped double quotes</
p>:";
myField_txt.htmlText = '<p align="left">This uses outer single quotes</p>';
myField_txt.htmlText = '<p align=\\'left\\\'>This uses escaped single quotes</
p>';
```

Non è necessario anteporre il carattere di escape alle virgolette doppie se si carica del testo da un file esterno; è invece necessario se si assegna una stringa di testo in ActionScript.

Informazioni sui tag HTML supportati

In questa sezione sono elencati i tag HTML incorporati supportati da Flash Player. È possibile, inoltre, creare nuovi stili e tag tramite CSS. Vedere “[Formattazione del testo con gli stili CSS](#)” a pagina 461.

Per ulteriori informazioni sui tag HTML supportati, consultare i seguenti argomenti:

- “[Tag di ancoraggio](#)” a pagina 477
- “[Tag del grassetto](#)” a pagina 478
- “[Tag per l'interruzione di riga](#)” a pagina 478
- “[Tag del carattere](#)” a pagina 479
- “[Tag per le immagini](#)” a pagina 479
- “[Tag del corsivo](#)” a pagina 481
- “[Tag per le voci di elenco](#)” a pagina 481
- “[Tag del paragrafo](#)” a pagina 482
- “[Tag SPAN](#)” a pagina 482
- “[Tag TEXTFORMAT](#)” a pagina 483
- “[Tag di sottolineatura](#)” a pagina 484

Tag di ancoraggio

Il tag <a> crea un collegamento ipertestuale e supporta i seguenti attributi:

- **href** Una stringa della lunghezza massima di 128 caratteri che specifica l'URL della pagina da caricare nel browser. L'URL può essere assoluto o relativo alla posizione del file SWF che carica la pagina. Un esempio di riferimento assoluto a un URL è `http://www.macromedia.com`, mentre `/index.html` è un riferimento relativo.
- **target** Specifica il nome della finestra target in cui caricare la pagina. Le opzioni disponibili sono `_self`, `_blank`, `_parent` e `_top`. L'opzione `_self` specifica il frame corrente nella finestra corrente, `_blank` indica una nuova finestra, `_parent` specifica l'elemento principale del frame corrente, mentre `_top` indica il frame di primo livello nella finestra corrente.

Il codice HTML seguente, ad esempio, crea il collegamento "Go home" che apre la pagina `www.macromedia.com` in una nuova finestra del browser:

```
urlText_txt.htmlText = "<a href='http://www.macromedia.com' target='_blank'>Go home</a>";
```

È possibile utilizzare il protocollo speciale `asfunction` per fare in modo che il collegamento esegua una funzione ActionScript in un file SWF anziché aprire un URL. Per ulteriori informazioni sul protocollo `asfunction`, vedere `asfunction` protocol nella *Guida di riferimento di ActionScript 2.0*.

Per i tag di ancoraggio è possibile anche definire gli stili `a:link`, `a:hover` e `a:active` utilizzando i fogli di stile. Vedere “[Assegnazione dello stile a tag HTML incorporati](#)” a pagina 469.

NOTA

Gli URL assoluti devono essere preceduti dal prefisso `http://`, altrimenti Flash li considera come URL relativi.

Tag del grassetto

Il tag `` consente di eseguire il rendering del testo in grassetto, come nell'esempio seguente:

```
text3_txt.htmlText = "He was <b>ready</b> to leave!";
```

Affinché il testo possa essere visualizzato con il carattere utilizzato, deve essere disponibile un tipo di carattere grassetto.

Tag per l'interruzione di riga

Il tag `
` crea un'interruzione di riga nel campo di testo. Per poter utilizzare questo tag, il campo deve essere multiriga.

Nell'esempio seguente la riga viene interrotta tra le due frasi:

```
this.createTextField("text1_txt", 1, 10, 10, 200, 100);
text1_txt.html = true;
text1_txt.multiline = true;
text1_txt.htmlText = "The boy put on his coat.<br />His coat was <font
color='#FF0033'>red</font> plaid.";
```

Tag del carattere

Il tag `` specifica un tipo di carattere o un elenco di tipi di carattere per la visualizzazione del testo.

Questo tag supporta i seguenti attributi:

- **color** Sono supportati solo valori esadecimali (#FFFFFF) relativi a colori. Il seguente codice HTML crea ad esempio testo in colore rosso:
`myText_txt.htmlText = "This is red text";`
- **face** Specifica il nome del carattere da utilizzare. Come illustrato nell'esempio seguente, è possibile anche specificare un elenco di nomi di carattere separati da virgola. In tal caso Flash Player sceglie il primo carattere disponibile:
`myText_txt.htmlText = "Displays as either Times or Times New Roman...";`

Se il carattere specificato non è installato nel computer dell'utente o non è incorporato nel file SWF, Flash Player sceglie un altro carattere.

Per ulteriori informazioni sull'incorporamento di caratteri nelle applicazioni Flash, vedere `embedFonts` (`TextField.embedFonts` property) nella *Guida di riferimento di ActionScript 2.0* e “Impostazione delle opzioni per il testo dinamico e di input” in *Uso di Flash*.

- **size** Specifica le dimensioni del carattere in pixel, come nell'esempio seguente:
`myText_txt.htmlText = "This is blue, 24-point text";`
- Anziché dimensioni in pixel, è possibile specificare dimensioni relative in punti, ad esempio +2 o -4.

Tag per le immagini

Il tag `` consente di incorporare file di immagine JPEG, GIF, PNG, SWF e clip filmato esterni in campi di testo e istanze di componenti `TextArea`. Il testo si dispone automaticamente attorno alle immagini incorporate nei campi di testo o nei componenti. Per utilizzare questo tag, impostare il campo di testo di input o dinamico come multiriga e consentire il ritorno a capo del testo.

Per creare un campo di testo multiriga con a capo automatico, effettuare una delle seguenti operazioni:

- Nell'ambiente di creazione Flash, selezionare un campo di testo sullo stage, quindi, nella finestra di ispezione Proprietà, selezionare Multiriga dal menu Tipo testo.
- Per un campo di testo creato in fase di runtime con `createTextField` (`MovieClip.createTextField` method), impostare le proprietà multiline (`TextField.multiline` property) e `multiline` (`TextField.multiline` property) dell'istanza del nuovo campo di testo su `true`.

Per il tag `` esiste un solo attributo obbligatorio, `src`, che specifica il percorso di un file di immagine o SWF oppure l'identificatore di concatenamento di un simbolo di clip filmato nella libreria. Tutti gli altri attributi sono facoltativi.

Il tag `` supporta i seguenti attributi:

- `src` Specifica l'URL di un file di immagine o SWF oppure l'identificatore di concatenamento di un simbolo della libreria relativo a un clip filmato. Questo è l'unico attributo obbligatorio, tutti gli altri sono facoltativi. La visualizzazione dei file esterni (JPEG, GIF, PNG e SWF) avviene solo dopo lo scaricamento completo dei file.
- `id` Specifica il nome dell'istanza relativa a un clip filmato, creato da Flash Player, che contiene il file di immagine o SWF o il clip filmato incorporato. Questo attributo è utile se si desidera controllare il contenuto incorporato con ActionScript.
- `width` La larghezza dell'immagine, del file SWF o del clip filmato inserito, in pixel.
- `height` L'altezza di un'immagine, di un file SWF o di un clip filmato in pixel.
- `align` Specifica l'allineamento orizzontale dell'immagine incorporata nel campo di testo. I valori accettati sono `left` e `right`. Il valore predefinito è `left`.
- `hspace` Specifica la quantità di spazio orizzontale intorno all'immagine in cui non viene visualizzato testo. Il valore predefinito è 8.
- `vspace` Specifica la quantità di spazio verticale intorno all'immagine in cui non viene visualizzato testo. Il valore predefinito è 8.

Per ulteriori informazioni ed esempi sull'uso del tag ``, vedere “[Informazioni sull'incorporamento di immagini, file SWF e clip filmato in campi di testo](#)” a pagina 486.

Tag del corsivo

Il tag `<i>` consente di visualizzare in corsivo il testo a cui viene applicato, come nel codice seguente:

That is very `<i>interesting</i>`.

Il rendering dell'esempio di codice avviene in questo modo:

That is very *interesting*.

Per il carattere utilizzato deve essere disponibile un tipo di carattere corsivo.

Tag per le voci di elenco

Il tag `` inserisce un punto elenco davanti al testo che racchiude, come illustrato nell'esempio seguente:

Grocery list:

```
<li>Apples</li>
<li>Oranges</li>
<li>Lemons</li>
```

Il rendering dell'esempio di codice avviene in questo modo:

Grocery list:

- Apples
- Oranges
- Lemons

NOTA

I tag `` e `` per gli elenchi con ordinamento e senza ordinamento non sono riconosciuti da Flash Player e quindi non modificano il rendering dell'elenco. Tutte le voci dell'elenco utilizzano i punti elenco.

Tag del paragrafo

Il tag `<p>` crea un nuovo paragrafo. Per poter utilizzare questo tag, il campo deve essere multiriga.

Il tag `<p>` supporta i seguenti attributi:

- `align` Specifica l'allineamento del testo del paragrafo; i valori validi sono `left`, `right`, `justify` e `center`.
- `class` Specifica una classe di stile CSS definita da un oggetto `TextField.StyleSheet`. Per ulteriori informazioni, consultare “[Uso delle classi di stile](#)” a pagina 468.

Nell'esempio seguente viene utilizzato l'attributo `align` per allineare il testo a destra del campo di testo.

```
this.createTextField("myText_txt", 1, 10, 10, 400, 100);
myText_txt.html = true;
myText_txt.multiline = true;
myText_txt.htmlText = "<p align='right'>This text is aligned on the
right side of the text field</p>";
```

Nell'esempio seguente viene utilizzato l'attributo `class` per assegnare una classe di stile di testo al tag `<p>`.

```
var myStyleSheet:TextField.StyleSheet = new TextField.StyleSheet();
myStyleSheet.setStyle(".blue", {color:'#99CCFF', fontSize:18});
this.createTextField("test_txt", 10, 0, 0, 300, 100);
test_txt.html = true;
test_txt.styleSheet = myStyleSheet;
test_txt.htmlText = "<p class='blue'>This is some body-styled text.</
p>.";
```

Tag SPAN

Il tag `` può essere usato solo con stili di testo CSS. Per ulteriori informazioni, consultare “[Formattazione del testo con gli stili CSS](#)” a pagina 461. Questo tag supporta i seguenti attributi:

- `class` Specifica una classe di stile CSS definita da un oggetto `TextField.StyleSheet`. Per ulteriori informazioni sulla creazione delle classi di stile, vedere “[Uso delle classi di stile](#)” a pagina 468.

Tag TEXTFORMAT

Il tag <textformat> permette di utilizzare nei campi di testo HTML un sottoinsieme delle proprietà di formattazione dei paragrafi appartenente alla classe TextFormat, tra cui interlinea, rientro, margini e spazi di tabulazione. È possibile combinare tag <textformat> con i tag HTML incorporati.

Il tag <textformat> ha i seguenti attributi:

- **blockindent** Specifica il rientro del blocco di testo in punti; corrisponde a TextFormat.blockIndent. Vedere blockIndent (TextFormat.blockIndent property) nella *Guida di riferimento di ActionScript 2.0*.
- **indent** Specifica il rientro dal margine sinistro del primo carattere del paragrafo; corrisponde a TextFormat.indent. È consentito l'utilizzo di interi negativi, vedere indent (TextFormat.indent property) nella *Guida di riferimento di ActionScript 2.0*.
- **leading** Specifica l'interlinea (spazio verticale) tra le righe; corrisponde a TextFormat.leading. È consentito l'utilizzo di interi negativi, vedere leading (TextFormat.leading property) nella *Guida di riferimento di ActionScript 2.0*.
- **leftmargin** Specifica il margine sinistro del paragrafo in punti; corrisponde a TextFormat.leftMargin. Vedere leftMargin (TextFormat.leftMargin property) nella *Guida di riferimento di ActionScript 2.0*.
- **rightmargin** Specifica il margine destro del paragrafo in punti; corrisponde a TextFormat.rightMargin. Vedere rightMargin (TextFormat.rightMargin property) nella *Guida di riferimento di ActionScript 2.0*.
- **tabstops** Specifica gli spazi di tabulazione personalizzati sotto forma di un array di numeri interi non negativi; corrisponde a TextFormat.tabStops. Vedere tabStops (TextFormat.tabStops property) nella *Guida di riferimento di ActionScript 2.0*.

La tabella di dati di seguito, con le intestazioni di riga in grassetto, è il risultato dell'esempio di codice nella seguente procedura:

Name	Age	Occupation
Rick	33	Detective
AJ	34	Detective

Per creare una tabella di dati formattata con spazi di tabulazione:

1. Creare un nuovo documento Flash e salvarlo come **tabstops.fla**.
2. Nella linea temporale, selezionare il primo fotogramma sul livello 1.
3. Aprire il pannello Azioni (Finestra > Azioni) e immettere il seguente codice nel pannello:

```
// Crea un nuovo campo di testo.  
this.createTextField("table_txt", 99, 50, 50, 450, 100);  
table_txt.multiline = true;  
table_txt.html = true;  
// Crea intestazioni di colonna formattate in grassetto e separate da  
// tabulazioni.  
var rowHeaders:String = "<b>Name\tAge\tOccupation</b>";  
  
// Crea righe con dati.  
var row_1:String = "Rick\t33\tDetective";  
var row_2:String = "AJ\t34\tDetective";  
  
// Imposta due spazi di tabulazione a 50 e 100 punti.  
table_txt.htmlText = "<textformat tabstops='[50,100]'>";  
table_txt.htmlText += rowHeaders;  
table_txt.htmlText += row_1;  
table_txt.htmlText += row_2;  
table_txt.htmlText += "</textformat>";
```

La sequenza di escape dei caratteri di tabulazione (\t) consente di aggiungere tabulazioni tra ogni colonna della tabella. Per aggiungere del testo, utilizzare l'operatore +=.

4. Selezionare Controllo > Prova filmato per visualizzare la tabella formattata.

Tag di sottolineatura

Il tag <u> consente di sottolineare il testo a cui viene applicato, come nel codice seguente:

This is <u>underlined</u> text.

Il rendering del codice avviene in questo modo:

This is underlined text.

Informazioni sulle entità HTML

Le entità HTML consentono di visualizzare determinati caratteri in campi di testo in formato HTML in modo che non siano interpretati come testo HTML. Ad esempio, i caratteri minore di (<) e maggiore di (>) vengono utilizzati per racchiudere tag HTML, quali e . In Flash, per visualizzare carattere maggiore di o minore di in campi di testo in formato HTML, è necessario sostituire a tali caratteri entità HTML. Il seguente codice ActionScript crea un campo di testo in formato HTML sullo stage e utilizza entità HTML per visualizzare la stringa “” senza che il testo sia visualizzato in grassetto:

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "The &lt;b&gt; tag makes text appear <b>bold</b>.";
```

In fase di runtime, il codice visualizza in Flash il seguente testo sullo stage:

The tag makes text appear bold.

Oltre ai simboli maggiore di e minore di, Flash riconosce altre entità HTML, elencate nella seguente tabella.

Entità	Descrizione
<	< (minore di)
>	> (maggior di)
&	& (e commerciale)
"	" (virgolette doppie)
'	' (apostrofo, virgolette singole)

Flash supporta inoltre codici di caratteri esplicativi, quali ‘ (e commerciale- ASCII) e ‘ (e commerciale - Unicode).

Il seguente codice ActionScript illustra l'uso di codici di carattere ASCII o Unicode per incorporare un carattere tilde (~):

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "&#126;"; // tilde (ASCII)
my_txt.htmlText += "\t"
my_txt.htmlText += "&#x007E;"; // tilde (Unicode)
```

Informazioni sull'incorporamento di immagini, file SWF e clip filmato in campi di testo

In Flash Player 7 e nelle versioni successive, il tag `` può essere utilizzato per incorporare file JPEG, GIF, PNG e SWF e clip filmato all'interno di campi di testo dinamici e di input e di istanze di componenti TextArea. Per un elenco completo degli attributi del tag ``, vedere “[Tag per le immagini](#)” a pagina [479](#).

Flash visualizza il contenuto multimediale incorporato in un campo di testo alle dimensioni reali. Per specificare le dimensioni dei contenuti multimediali incorporati, utilizzare gli attributi `height` e `width` del tag ``. Vedere “[Informazioni sull'indicazione dei valori di altezza e larghezza](#)” a pagina [488](#).

In genere, un'immagine incorporata in un campo di testo compare nella riga dopo il tag ``. Tuttavia, quando il tag `` è il primo carattere del campo di testo, l'immagine è visualizzata nella prima riga del campo di testo.

Incorporamento di file di immagini e SWF

Per incorporare un file di immagine o SWF in un campo di testo, specificare il percorso relativo o assoluto corrispondente al file di immagine (GIF, JPEG, or PNG) o SWF per l'attributo `src` del tag ``. Ad esempio, il seguente codice inserisce un file GIF memorizzato nella stessa directory del file SWF (un indirizzo relativo in linea o fuori linea).

Incorporamento di un'immagine in un campo di testo:

1. Creare un nuovo documento Flash e salvarlo come `embedding.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("image1_txt", 10, 50, 50, 450, 150);
image1_txt.html = true;
image1_txt.htmlText = "<p>Here's a picture from my vacation:<img
src='beach.gif'>";
```

Il codice crea un nuovo campo di testo dinamico sullo stage, abilita il formato HTML e aggiunge del testo e un'immagine locale al campo di testo.

3. Aggiungere il codice ActionScript seguente dopo il codice creato al punto precedente:

```
this.createTextField("image2_txt", 20, 50, 200, 400, 150);
image2_txt.html = true;
image2_txt.htmlText = "<p>Here's a picture from my garden:<img
src='http://www.helpexamples.com/flash/images/image2.jpg'>";
```

È inoltre possibile inserire un'immagine utilizzando un indirizzo assoluto. Il codice precedente inserisce un file JPEG memorizzato in una directory su un server. Il file SWF che contiene questo codice può trovarsi sul disco rigido locale o su un server.

4. Salvare il documento e selezionare Controllo > Prova filmato per provare il documento.

Il campo di testo superiore contiene una frase e molto probabilmente un messaggio di errore nel pannello Output, indica che Flash non è stato in grado di individuare un file chiamato beach.gif nella directory corrente. Il campo di testo inferiore contiene una frase e l'immagine di un fiore caricata dal server remoto.

Copiare un'immagine GIF nella stessa directory del file FLA, rinominare l'immagine come beach.gif e selezionare Controllo > Prova filmato per provare nuovamente il documento Flash.



Quando si utilizzano URL assoluti, verificare che all'URL sia anteposto il prefisso `http://`.

Incorporamento di simboli di clip filmato

Per incorporare il simbolo di un clip filmato in un campo di testo, occorre specificare l'identificatore di concatenamento del simbolo relativo all'attributo `<src>` del tag `img`. Per informazioni sulla definizione di un identificatore di concatenamento, vedere [“Associazione di un simbolo clip filmato allo stage” a pagina 394](#).

Il seguente codice inserisce ad esempio un simbolo di un clip filmato con l'identificatore di concatenamento `symbol_ID` in un campo di testo dinamico con il nome di istanza `textField_txt`.

Per incorporare un clip filmato in un campo di testo:

1. Creare un nuovo documento Flash e salvarlo come `embeddedmc.fla`.
2. Disegnare una nuova forma sullo stage, oppure selezionare File > Importa > Importa nello stage e selezionare con dimensioni di circa 100 x 100 pixel.
3. Convertire la forma o l'immagine importata nel punto precedente selezionandola sullo stage e premendo il tasto F8 per aprire la finestra di dialogo Converti in simbolo.
4. Impostare il comportamento su Movie Clip e inserire un nome di simbolo descrittivo. Selezionare il quadrato superiore sinistro della griglia dei punti di registrazione e fare clic su Avanzato per passare alla modalità avanzata, se non è già selezionata.
5. Selezionare Esporta per ActionScript ed Esporta nel primo fotogramma.
6. Inserire l'identificatore di concatenamento `img_id` nella casella di testo Identificatore e fare clic su OK.

- 7.** Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "<p>Here's a movie clip symbol:<img
src='img_id'>";
```

Per una corretta e completa visualizzazione di un clip filmato incorporato, il punto di registrazione del simbolo corrispondente deve essere 0,0.

- 8.** Salvare le modifiche apportate al documento Flash.
9. Selezionare Controllo > Prova filmato per provare il documento Flash.

Informazioni sull'indicazione dei valori di altezza e larghezza

Se si specificano gli attributi `width` e `height` di un tag ``, nel campo di testo viene riservato dello spazio per il file di immagine, il file SWF o il clip filmato. Al termine dello scaricamento di un file di immagine o SWF, il file viene visualizzato nello spazio riservato. Flash modifica le dimensioni dei media in base ai valori stabiliti per gli attributi `height` e `width`. Per ridimensionare l'immagine è necessario immettere valori per entrambi gli attributi, `height` e `width`.

Se non si specificano valori per gli attributi `height` e `width`, non viene riservato spazio per il contenuto multimediale incorporato. Dopo aver completato lo scaricamento di un file di immagine o SWF, Flash inserisce tale oggetto nel campo di testo mantenendone inalterate le dimensioni, ma modificando la disposizione del testo intorno all'oggetto.



Se le immagini vengono caricate in modo dinamico in un campo di testo che contiene testo, si consiglia di specificare la larghezza e l'altezza dell'immagine originale, affinché il testo si disponga correttamente attorno allo spazio riservato all'immagine.

Controllo di contenuti multimediali incorporati con ActionScript

Flash crea un nuovo clip filmato per ogni tag `` e incorpora tale clip nell'oggetto `TextField`. L'attributo `id` del tag `` consente di assegnare un nome di istanza al clip filmato creato. Ciò consente di controllare il clip filmato con ActionScript.

Il clip filmato creato da Flash viene aggiunto come clip filmato secondario al campo di testo contenente l'immagine.

Il seguente esempio incorpora un file SWF in un campo di testo.

Per incorporare un file SWF in un campo di testo:

1. Creare un nuovo documento Flash.
2. Ridimensionare le dimensioni dello stage del documento su 100 x 100 pixel.
3. Usare lo strumento Rettangolo e disegnare un quadrato rosso sullo stage.
4. Ridimensionare il quadrato su 80 x 80 pixel utilizzando la finestra di ispezione Proprietà, quindi spostare la figura al centro dello stage.
5. Selezionare il fotogramma 20 sulla linea temporale e premere F7 (Windows o Macintosh) per inserire un nuovo fotogramma chiave vuoto.
6. Utilizzando lo strumento Ovale, disegnare un cerchio blu sullo stage, nel fotogramma 20.
7. Ridimensionare il cerchio su 80 x 80 pixel utilizzando la finestra di ispezione Proprietà, quindi spostarlo al centro dello stage.
8. Selezionare un fotogramma vuoto tra i fotogrammi 1 e 20, quindi impostare il tipo di interpolazione su Forma nella finestra di ispezione Proprietà.
9. Salvare il documento come **animation.fla**
10. Selezionare Controllo > Prova filmato per visualizzare in anteprima l'animazione.

Il file SWF viene creato nella stessa directory del file FLA. Affinché l'esercizio funzioni correttamente, è necessario che sia generato il file SWF in modo da poterlo caricare in un file FLA separato.

11. Creare un nuovo file FLA e salvarlo come **animationholder.fla**.
Salvare il file nella stessa cartella del file animation.fla creato in precedenza.
12. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "Here's an interesting animation: <img
src='animation.swf' id='animation_mc'>";
```

In questo caso, il percorso completo del nuovo clip filmato creato è
textField_txt.animation_mc.

13. Salvare le modifiche apportate al documento Flash e selezionare Controllo > Prova filmato per visualizzare in anteprima l'animazione nel campo di testo.

Per controllare la riproduzione del file SWF in un campo di testo, completare l'esercizio seguente.

Per controllare un file SWF riprodotto in un campo di testo:

1. Completare i passaggi della prima procedura in “[Controllo di contenuti multimediali incorporati con ActionScript](#)” a pagina 488.
2. Creare un'istanza di pulsante sullo stage e assegnarvi il nome di istanza **stop_btn** nella finestra di ispezione Proprietà.
3. Aggiungere il codice ActionScript seguente dopo il codice esistente nel fotogramma 1 della linea temporale principale:

```
stop_btn.onRelease = function() {
    textField_txt.animation_mc.stop();
};
```
4. Selezionare Controllo > Prova filmato per provare l'applicazione.

Ogni qualvolta si fa clic sull'istanza di pulsante **stop_btn**, si ferma la linea temporale dell'animazione nidificata nel campo di testo.

Per ulteriori informazioni sull'uso di contenuti multimediali incorporati come collegamenti ipertestuali, vedere “[Informazioni sull'uso di contenuti multimediali incorporati come collegamenti ipertestuali](#)” a pagina 490.

Informazioni sull'uso di contenuti multimediali incorporati come collegamenti ipertestuali

Per utilizzare un file di immagine o SWF oppure un clip filmato come collegamento ipertestuale, inserire il tag `` all'interno di un tag `<a>`:

```
textField_txt.htmlText = "Click the image to return home<a
    href='home.htm'><img src='home.jpg'></a>";
```

Quando il puntatore del mouse viene posizionato su un'immagine, un file SWF o un clip filmato racchiuso tra tag `<a>`, assume la forma di una mano, analogamente a quanto avviene con i normali collegamenti ipertestuali. Le operazioni interattive, quali i clic del mouse e la pressione di tasti, non vengono registrate nei file SWF e nei clip filmato racchiusi tra tag `<a>`.

Per informazioni sull'incorporamento di contenuti multimediali, vedere “[Informazioni sull'uso di contenuti multimediali incorporati come collegamenti ipertestuali](#)” a pagina 490.

Esempio: Creazione di testo scorrevole

Flash fornisce vari modi per creare testo scorrevole. È possibile rendere scorrevoli i campi di testo di input e dinamici selezionando l'opzione Scorrrevole dal menu Testo o dal menu di scelta rapida, oppure facendo doppio clic sulla maniglia del blocco di testo mentre si preme il tasto Maiusc.

È possibile utilizzare le proprietà `scroll` e `maxscroll` dell'oggetto `TextField` per controllare lo scorrimento verticale e le proprietà `hscroll` e `maxhscroll` per controllare lo scorrimento orizzontale di un campo di testo. Le proprietà `scroll` e `hscroll` specificano rispettivamente le posizioni di scorrimento in verticale e orizzontale; tali proprietà possono essere lette e scritte. Le proprietà `maxscroll` e `maxhscroll` specificano rispettivamente le posizioni di scorrimento massime in verticale e orizzontale; tali proprietà sono di sola lettura.

Il componente `TextArea` consente di creare campi di testo scorrevoli in modo semplice e scrivendo una quantità di codice minima. Per ulteriori informazioni, vedere “Componente `TextArea`” nella *Guida di riferimento dei componenti*.

Per creare un campo di testo dinamico scorrevole:

Effettuare una delle seguenti operazioni:

- Fare doppio clic sulla maniglia del campo di testo dinamico tenendo contemporaneamente premuto il tasto Maiusc.
- Selezionare il campo di testo dinamico con lo strumento Selezione, quindi scegliere Testo > Scorrrevole.
- Selezionare il campo di testo dinamico con lo strumento Selezione. Fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) sul campo di testo dinamico, quindi selezionare Testo > Scorrrevole.

Per utilizzare la proprietà `scroll` per creare testo scorrevole:

1. Effettuare una delle seguenti operazioni:

- Usare lo strumento Testo per trascinare un campo di testo sullo stage. Assegnare al campo di testo il nome di istanza `textField_txt` nella finestra di ispezione Proprietà.
- Usare ActionScript per creare un campo di testo in modo dinamico con il metodo `MovieClip.createTextField()`. Assegnare al campo di testo il nome di istanza `textField_txt` come parametro del metodo.



Se il testo *non* viene caricato all'interno del file SWF in modo dinamico, selezionare Testo > Scorrrevole dal menu principale.

- 2.** Creare i pulsanti per effettuare lo scorrimento verso l'alto e verso il basso oppure scegliere Finestra > Librerie comuni > Pulsanti e trascinare i pulsanti sullo stage.

È possibile utilizzare tali pulsanti per scorrere il testo verso l'alto o verso il basso.

- 3.** Selezionare il pulsante Giù sullo stage e digitare **down_btn** nella casella di testo Nome istanza.

- 4.** Selezionare il pulsante Su sullo stage e digitare **up_btn** nella casella di testo Nome istanza.

- 5.** Selezionare il fotogramma 1 nella linea temporale e nel pannello Azioni (Finestra > Azioni) immettere il seguente codice per scorrere il testo verso il basso nel campo di testo:

```
down_btn.onPress = function() {  
    textField_txt.scroll += 1;  
};
```

- 6.** Dopo il codice ActionScript riportato al punto 5, immettere il seguente codice per scorrere il testo verso l'alto:

```
up_btn.onPress = function() {  
    textField_txt.scroll -= 1;  
};
```

Mediante i pulsanti Su e Giù, è possibile scorrere qualunque testo caricato nel campo di testo `textField_txt`.

Informazioni sulle stringhe e sulla classe String

Nella programmazione, una stringa è una serie ordinata di caratteri. Nei documenti Flash e nei file di classe si utilizzano spesso delle stringhe per visualizzare il testo nelle applicazioni, ad esempio nei campi di testo. È inoltre possibile memorizzare valori come stringhe, che possono quindi essere utilizzate in un'applicazione per vari scopi. Le stringhe possono essere inserite direttamente nel codice ActionScript, racchiudendo i caratteri dei dati tra virgolette. Per ulteriori informazioni sulla creazione di stringhe, vedere “[Creazione di stringhe](#) a pagina 501. Per informazioni sull'uso dei campi di testo, vedere “[Uso della classe TextField](#)” a pagina 418.

Ogni carattere può essere associato a un codice di carattere specifico, che è inoltre possibile utilizzare facoltativamente per visualizzare del testo. Ad esempio, il carattere “A” è rappresentato dal codice di carattere Unicode 0041 o dal codice ASCII (American Standard Code for Information Interchange) 65. Per ulteriori informazioni sui codici di caratteri e sulle tabelle di codici, vedere www.unicode.org/charts. Il modo di rappresentare le stringhe in un documento Flash dipende, come si può vedere, in larga misura dal set di caratteri scelto e dalla modalità di codifica dei caratteri.

Per *codifica dei caratteri* si intende il codice, o metodo, utilizzato per rappresentare un set di caratteri di un linguaggio in codici rappresentativi, ad esempio valori numerici. Il *codice di carattere* (come detto nel paragrafo precedente) è la tabella dei valori mappati, ad esempio una tabella ASCII dove A è uguale a 65, che vengono decrittografati dal metodo di codifica in un programma nel computer.

Ogni lettera, ad esempio, nella lingua inglese ha un corrispondente codice numerico in una codifica dei caratteri. ASCII codifica ogni lettera, numero e alcuni simboli in versioni binarie a 7 bit di ogni numero intero. ASCII è un set di caratteri costituito da 95 caratteri stampabili e numerosi caratteri di controllo; i computer lo utilizzano per rappresentare il testo.

Anche *Unicode*, come ASCII, è un modo di associare un codice con ogni lettera dell'alfabeto. ASCII non è in grado di supportare set di caratteri estesi, come il cinese, quindi Unicode è uno standard molto utile per la codifica di alcune lingue. Unicode è lo standard per i set di caratteri che possono rappresentare qualsiasi lingua. Questo standard è stato studiato per facilitare lo sviluppo in più lingue. Il codice del carattere specifica il carattere rappresentato e lo standard tenta di fornire un metodo universale per la codifica dei caratteri che fanno parte di qualsiasi lingua, in modo da consentire la visualizzazione delle stringhe su qualsiasi sistema informatico, piattaforma o software. È quindi compito del programma utilizzato, ad esempio Flash o un browser Web, visualizzare il glifo del carattere, ovvero il suo aspetto visivo.

Il numero di caratteri supportati da Unicode è stato ampliato nel corso degli anni, aggiungendovi il supporto per altre lingue più estese. Le codifiche dei caratteri sono denominate UTF (Unicode Transformation Format) e UCS (Universal Character Set), e comprendono UTF-8, UTF-16 e UTF-32. I numeri nella codifica UTF rappresentano il numero di bit in un'unità, mentre il numero in una codifica UCS rappresenta i byte.

- UTF-8 è la codifica standard per lo scambio di testo, ad esempio i sistemi di posta elettronica. UTF è un sistema a 8 bit.
- UTF-16 viene comunemente utilizzato per l'elaborazione interna.

Le stringhe nelle applicazioni possono avere lunghezze diverse. È possibile determinare la lunghezza di una stringa, anche se può variare in base alla lingua utilizzata. Alla fine di una stringa può inoltre essere presente un carattere di terminazione; questo carattere nullo non ha alcun valore. Il carattere di terminazione non è un vero carattere, ma può essere utilizzato per determinare la fine di una stringa. Ad esempio, se si utilizzano connessioni socket, è possibile cercare il carattere di terminazione per individuare la fine di una stringa, come in un programma chat.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, strings.fla. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings*.

Per ulteriori informazioni sulle stringhe e sulla classe String, consultare i seguenti argomenti:

- “[Informazioni sul pannello Stringhe](#)” a pagina 494
- “[Uso della classe Locale](#)” a pagina 495
- “[Uso di un editor di metodo di input](#)” a pagina 497
- “[Informazioni sulla classe String](#)” a pagina 500
- “[Creazione di stringhe](#)” a pagina 501
- “[Informazioni sul carattere escape](#)” a pagina 503
- “[Analisi e confronto dei caratteri nelle stringhe](#)” a pagina 504
- “[Conversione e concatenazione di stringhe](#)” a pagina 507
- “[Restituzione di sottostringhe](#)” a pagina 511

Informazioni sul pannello Stringhe

Il pannello Stringhe consente di creare e aggiornare il contenuto in più lingue. È possibile specificare il contenuto di campi di testo in più lingue e fare in modo che Flash determini automaticamente il contenuto da visualizzare in una determinata lingua in base alla lingua del computer su cui viene eseguito Flash Player.

Per informazioni generali sul pannello Stringhe e sul suo utilizzo nelle applicazioni, consultare i seguenti argomenti in *Uso di Flash*:

- “[Creazione di testo in più lingue mediante il pannello Stringhe](#)” a pagina 423
- “[Informazioni sulla modifica delle stringhe nel pannello Stringhe](#)” a pagina 427
- “[Traduzione del testo nel pannello Stringhe o in un file XML](#)” a pagina 432
- “[Importazione di un file XML nel pannello Stringhe](#)” a pagina 433

Per controllare la visualizzazione di testo in più lingue è possibile utilizzare la classe Locale. Per ulteriori informazioni, vedere “[Uso della classe Locale](#)” a pagina 495 e `Locale` (`mx.lang.Locale`) nella *Guida di riferimento di ActionScript 2.0*.

Uso della classe Locale

La classe Locale (mx.lang.Locale) consente di controllare il modo in cui il testo multilingua viene visualizzato in un'applicazione Flash in fase di runtime. Con il pannello Stringhe è possibile utilizzare identificatori di stringa in luogo di valori letterali di stringa in campi di testo dinamici; in questo modo è possibile creare un file SWF che visualizza il testo caricato da un file XML specifico di una lingua. Per visualizzare le stringhe specifiche di una lingua contenute nei file XLIFF (XML Localization Interchange File Format) è possibile utilizzare i seguenti metodi.

automaticamente in fase di runtime Flash Player sostituisce gli identificatori di stringa con le stringhe del file XML che corrisponde al codice di lingua di sistema predefinito restituito da `language` (`capabilities.language` property).

manualmente mediante la lingua dello stage Gli identificatori di stringa vengono sostituiti dalle stringhe in fase di compilazione del file SWF e non possono essere modificati da Flash Player.

mediante ActionScript in fase di runtime È possibile controllare la sostituzione di identificatori di stringa mediante ActionScript, controllato in fase di runtime. Questa opzione offre il controllo sia sui tempi che sulla lingua della sostituzione degli identificatori di stringa. I metodi e le proprietà della classe Locale possono essere utilizzati per sostituire identificatori di stringa mediante ActionScript per controllare l'applicazione durante la riproduzione in Flash Player. Un esempio di utilizzo della classe Locale si trova nella seguente procedura.

Per utilizzare la classe Locale per creare siti multilingua:

1. Creare un nuovo documento Flash e salvarlo come `locale.fla`.
 2. Selezionare Finestra > Altri pannelli > Stringhe per aprire il pannello Stringhe, quindi fare clic su Impostazioni.
 3. Selezionare due lingue, en (inglese) e fr (francese), quindi fare clic su Aggiungi per inserire le lingue nel riquadro Lingue attive.
 4. Selezionare l'opzione mediante ActionScript in fase di runtime, impostare la lingua di runtime predefinita su francese, quindi fare clic su OK.
 5. Trascinare un componente ComboBox nello stage dalla cartella User Interface del pannello Componenti (Finestra > Componenti) e assegnarvi il nome di istanza `lang_cb`.
 6. Con lo strumento Testo creare un campo di testo dinamico sullo stage e darvi il nome di istanza `greeting_txt`.
 7. Con il campo di testo selezionato sullo stage, digitare un identificatore di stringa `greeting` nella casella di testo ID del pannello Stringhe, quindi fare clic su Applica.
- Flash converte la stringa greeting in IDS_GREETING.

8. Nella griglia del pannello Stringhe, immettere la stringa **hello** nella colonna en.

9. Nella colonna fr immettere la stringa **bonjour**.

Le stringhe vengono utilizzate con la casella combinata lang_cb per cambiare la lingua sullo stage.

10. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);
greeting_txt.autoSize = "left";
Locale.loadLanguageXML(lang_cb.value);

function langListener(eventObj:Object):Void {
    Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
    if (success) {
        greeting_txt.text = Locale.loadString("IDS_GREETING");
    } else {
        greeting_txt.text = "unable to load language XML file.";
    }
}
```

Il codice precedente ActionScript viene suddiviso in due sezioni. La prima sezione di codice importa la classe Locale e specifica un callback del listener che viene chiamato ogni qualvolta termina il caricamento di un file XML di lingua. In seguito la casella combinata lang_cb viene compilata con un array ordinato delle lingue disponibili. Ogni qualvolta cambia il valore lang_cb, il dispatcher di eventi di Flash attiva la funzione langListener(), che carica il file XML specifico della lingua. La seconda sezione di codice definisce due funzioni, langListener() e localeListener(). La prima, langListener(), viene chiamata ogni qualvolta l'utente modifica il valore della casella combinata lang_cb. La seconda funzione, localeListener(), viene chiamata ogni qualvolta termina il caricamento di un file di lingua XML. La funzione controlla il buon esito del caricamento e, in caso positivo, imposta la proprietà text dell'istanza greeting_txt sul saluto nella lingua selezionata.

11. Selezionare Controllo > Prova filmato per provare il documento Flash.

SUGGERIMENTO

Il file XML usato deve utilizzare il formato XML Localization Interchange File Format (XLIFF).

ATTENZIONE

Nella Guida di riferimento di ActionScript 2.0, la classe Locale differisce dalle altre classi, poiché non fa parte di Flash Player. Poiché la classe è installata nel percorso di classe dello strumento di creazione di Flash, viene compilata automaticamente nei file SWF. L'utilizzo della classe Locale aumenta leggermente le dimensioni del file SWF, poiché deve essere compilata nel file SWF.

Per ulteriori informazioni, vedere `Locale` (`mx.lang.Locale`) nella *Guida di riferimento di ActionScript 2.0*.

Uso di un editor di metodo di input

Gli editor del metodo di input (IME, Input Method Editor) consentono all'utente di digitare caratteri di testo non ASCII nelle lingue asiatiche quali il cinese, il giapponese e il coreano. In ActionScript la classe IME consente di manipolare direttamente l'IME del sistema operativo all'interno dell'applicazione Flash Player in esecuzione su un computer client.

Utilizzando ActionScript è possibile determinare quanto segue:

- Se sul computer di un utente è installato un IME.
- Se sul computer di un utente l'IME è abilitato o disabilitato.
- La modalità di conversione utilizzata dall'IME corrente.

La classe IME è in grado di determinare la modalità di conversione utilizzata dall'IME corrente: ad esempio, se è attivo l'IME giapponese, è possibile determinare se la modalità di conversione è Hiragana, Katakana (e così via) utilizzando il metodo `System.IME.getConversionMode()`. La modalità di conversione può essere impostata con il metodo `System.IME.setConversionMode()`.

NOTA

Attualmente non è possibile stabilire quale IME è attivo (se presente), o passare da un IME a un altro (ad esempio da inglese a giapponese o da coreano a cinese)

È anche possibile abilitare o disabilitare l'IME utilizzando l'applicazione in fase di runtime ed eseguire altre funzioni, a seconda del sistema operativo dell'utente. È possibile controllare se un sistema ha un IME mediante la proprietà `System.capabilities.hasIME`. L'esempio seguente illustra come determinare se l'utente ha un IME installato e attivo.

Per determinare se l'utente ha un IME installato e attivo:

1. Creare un nuovo documento Flash e salvarlo come `ime.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
if (System.capabilities.hasIME) {  
    if (System.IME.getEnabled()) {  
        trace("You have an IME installed and enabled.");  
    } else {  
        trace("You have an IME installed but not enabled.");  
    }  
} else {  
    trace("Please install an IME and try again.");  
}
```

Il codice controlla se il sistema corrente ha un IME installato; se ne trova uno, Flash controlla se è abilitato.

3. Selezionare Controllo > Prova filmato per provare il documento.

Nel pannello Output viene visualizzato un messaggio che indica se è installato un IME e se è attivo.

Per abilitare e disabilitare l'IME in Flash in fase di runtime è anche possibile utilizzare la classe IME. Il seguente esempio richiede che nel sistema sia installato un IME. Per ulteriori informazioni sull'installazione di un IME su una piattaforma specifica, consultare i seguenti collegamenti:

- www.microsoft.com/globaldev/default.mspx
- <http://developer.apple.com/documentation/>
- <http://java.sun.com>

È possibile abilitare e disabilitare un IME durante la riproduzione di un file SWF, come illustrato nel seguente esempio.

Per abilitare e disabilitare un editor di metodo di input in fase di runtime:

1. Creare un nuovo documento Flash e salvarlo come **ime2.fla**.
2. Creare due istanze di simbolo pulsante sullo stage e assegnarvi i nomi di istanza **enable_btn** e **disable_btn**.
3. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
checkIME();

var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_sans";

this.createTextField("ime_txt", 10, 100, 10, 320, 240);
ime_txt.border = true;
ime_txt.multiline = true;
ime_txt.setNewTextFormat(my_fmt);
ime_txt.type = "input";
ime_txt.wordWrap = true;

enable_btn.onRelease = function() {
    System.IME.setEnabled(true);
};

disable_btn.onRelease = function() {
    System.IME.setEnabled(false);
};

function checkIME():Boolean {
    if (System.capabilities.hasIME) {
        if (System.IME.getEnabled()) {
            trace("You have an IME installed and enabled.");
            return true;
        } else {
            trace("You have an IME installed but not enabled.");
            return false;
        }
    } else {
        trace("Please install an IME and try again.");
        return false;
    }
}
```

Il codice precedente viene suddiviso in cinque sezioni. La prima sezione chiama il metodo `checkIME()`, che visualizza un messaggio nel pannello Output se il sistema ha un IME installato o attivo. La seconda sezione definisce un oggetto formato di testo, che imposta il carattere su `_sans`. La terza sezione crea un campo di testo di input e applica il formato di testo personalizzato. La quarta sezione crea alcuni gestori di eventi per le istanze `enable_btn` e `disable_btn` create in un punto precedente. La quinta e ultima sezione di codice definisce la funzione `checkIME()` personalizzata, che controlla se il sistema corrente ha un IME installato e, in caso positivo, se l'IME è attivo.

4. Salvare il file FLA e selezionare Controllo > Prova filmato per provare il documento.



Questo esempio richiede che nel sistema sia installato un IME. Per informazioni sull'installazione di un IME, consultare i collegamenti indicati prima dell'esempio.

Immettere del testo nel campo di testo di input sullo stage. Impostare l'IME su una lingua diversa e scrivere nuovamente nel campo di testo di input. Flash Player inserisce i caratteri utilizzando il nuovo IME. Facendo clic sul pulsante `disable_btn` sullo stage, Flash torna a utilizzare la lingua precedente e ignora le impostazioni di IME correnti.

Per ulteriori informazioni su `System.capabilities.hasIME`, vedere `hasIME` (`capabilities.hasIME` property) nella *Guida di riferimento di ActionScript 2.0*.

Informazioni sulla classe String

Una stringa è anche una classe e un tipo di dati nel linguaggio ActionScript di base. Il tipo di dati String rappresenta una sequenza di caratteri a 16 bit che può includere lettere, numeri e segni di punteggiatura. Le stringhe vengono memorizzate come caratteri Unicode, utilizzando il formato UTF-16. Un'operazione su un valore String restituisce una nuova istanza della stringa. Il valore predefinito di una variabile dichiarata con il tipo di dati String è `null`.

Per ulteriori informazioni su stringhe, dati e valori, consultare il [Capitolo 10, “Dati e tipi di dati”](#).

La classe String contiene i metodi che consentono di utilizzare le stringhe di testo. Le stringhe risultano importanti quando si utilizzano molti oggetti e i metodi descritti in questo capitolo sono utili per gestire le stringhe utilizzate in molti oggetti, ad esempio le istanze TextField, XML, ContextMenu e FileReference.

La classe String è un wrapper per il tipo di dati di base stringa e fornisce i metodi e le proprietà che consentono di manipolare i valori di stringa di base. È possibile convertire in stringa il valore di qualunque oggetto utilizzando la funzione `String()`. Tutti i metodi della classe String, a eccezione di `concat()`, `fromCharCode()`, `slice()` e `substr()`, sono generici; in altre parole, chiamano `toString()` prima di eseguire le proprie operazioni e possono essere utilizzati con altri oggetti non String.

Dal momento che tutti gli indici di stringa sono con base zero, l'indice dell'ultimo carattere di una qualunque stringa `myStr` è `myStr.length - 1`.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, `strings.fla`. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings*.

Creazione di stringhe

È possibile chiamare uno qualunque dei metodi della classe String utilizzando il metodo della funzione di costruzione `new String()` o un valore letterale di stringa. Se si specifica un carattere letterale di stringa, l'interprete ActionScript lo converte automaticamente in un oggetto String temporaneo, chiama il metodo e infine elimina l'oggetto String temporaneo. È possibile utilizzare anche la proprietà `String.length` con un carattere letterale di stringa.

È importante non confondere un *valore letterale* di stringa con un *oggetto String*. Per ulteriori informazioni sui valori letterali di stringa e sull'oggetto String, consultare il [Capitolo 4, "Informazioni sui valori letterali"](#) a pagina 94.

Nell'esempio seguente, la riga di codice crea il valore letterale di stringa `firstStr`. Per dichiarare un valore letterale di stringa, utilizzare le virgolette semplici diritte ('') o le virgolette doppie diritte ("").

Per creare e utilizzare stringhe:

1. Creare un nuovo documento Flash e salvarlo come **strings.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var firstStr:String = "foo";
var secondStr:String = new String("foo");
trace(firstStr == secondStr); // true
var thirdStr:String;
trace(thirdStr); // indefinito
```

Questo codice definisce tre oggetti String: uno utilizza un valore letterale di stringa, uno utilizza l'operatore `new` e uno senza un valore iniziale. Le stringhe possono essere confrontate con l'operatore di uguaglianza (`==`), come indicato nella terza riga di codice. Quando si fa riferimento a variabili, il tipo di dati viene specificato solo in fase di definizione della variabile.

3. Selezionare Controllo > Prova filmato per provare il documento.

Utilizzare sempre i valori letterali di stringa a meno che non sia espressamente necessario utilizzare un oggetto String. Per ulteriori informazioni sui valori letterali di stringa e sull'oggetto String, consultare il [Capitolo 4, “Informazioni sui valori letterali” a pagina 94](#).

Per utilizzare le virgolette semplici diritte (') o le virgolette doppie diritte ("') in un valore letterale di stringa, utilizzare il carattere barra rovesciata (\) come carattere di escape. Le due stringhe seguenti sono equivalenti:

```
var firstStr:String = "That's \"fine\"";
var secondStr:String = 'That\'s "fine"';
```

Per ulteriori informazioni sull'uso del carattere barra rovesciata nelle stringhe, vedere [“Informazioni sul carattere escape” a pagina 503](#).

Tenere presente che il carattere “virgola inglese” o il carattere “virgola speciale” non può essere utilizzato nel codice ActionScript; si tratta infatti di caratteri diversi dalle virgolette diritte (') e (") ammesse nel codice. Quando si incolla testo da un'altra origine in ActionScript, ad esempio dal Web o da un documento di Word, assicurarsi di utilizzare virgolette diritte.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, strings.fla. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Strings*.

Informazioni sul carattere escape

Per definire altri caratteri nei valori letterali di stringa, è possibile utilizzare il carattere barra rovesciata (\).

Sequenza di escape	Descrizione
\b	Il carattere barra rovesciata.
\f	Il carattere di avanzamento pagina.
\n	Il carattere di avanzamento riga.
\r	Il carattere di ritorno a capo.
\t	Il carattere Tab.
\unnnn	Il carattere Unicode con il codice di carattere specificato mediante il numero esadecimale <i>nnnn</i> . Il codice \u263a, ad esempio, corrisponde al carattere smile.
\xnn	Il carattere ASCII con il codice di carattere specificato mediante il numero esadecimale <i>nn</i> .
\'	Una virgoletta semplice.
\"	Una virgoletta doppia.
\\\	Un carattere barra rovesciata.

Per ulteriori informazioni sui valori letterali di stringa, consultare il [Capitolo 4, “Informazioni sui valori letterali” a pagina 94](#) e [“Creazione di stringhe” a pagina 501](#).

Analisi e confronto dei caratteri nelle stringhe

A ogni carattere di una stringa corrisponde una posizione di indice nella stringa (un numero intero). La posizione di indice del primo carattere è 0. Ad esempio, nella stringa `yellow` il carattere `y` si trova nella posizione 0 e il carattere `w` nella posizione 5.

Ogni stringa dispone di una proprietà `length` che corrisponde al numero di caratteri contenuti nella stringa:

```
var companyStr:String = "macromedia";
trace(companyStr.length); // 10
```

Una stringa vuota e una stringa null hanno entrambe una lunghezza zero:

```
var firstStr:String = new String();
trace(firstStr.length); // 0
```

```
var secondStr:String = "";
trace(secondStr.length); // 0
```

Se una stringa non contiene valori, la sua lunghezza è impostata come indefinita:

```
var thirdStr:String;
trace(thirdStr.length); // indefinita
```



Se la stringa contiene un carattere di byte nullo (\0), il valore di stringa viene troncato.

Per definire una stringa, è inoltre possibile utilizzare codici di caratteri. Per ulteriori informazioni su codici di caratteri e codifica dei caratteri, vedere “[Informazioni sulle stringhe e sulla classe String](#)” a pagina 492.

Il seguente esempio crea una variabile dal nome `myStr` e imposta il valore della stringa in base ai valori ASCII passati al metodo `String.fromCharCode()`:

```
var myStr:String =
    String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
trace(myStr); // hello world!
```

Ciascun numero elencato nel metodo `fromCharCode()` nel codice precedente rappresenta un singolo carattere. Ad esempio, il valore ASCII 104 rappresenta una `h` minuscola e il valore ASCII 32 rappresenta il carattere spazio.

È possibile utilizzare il metodo `String.fromCharCode()` anche per la conversione di valori Unicode, sebbene il valore unicode debba essere convertito da valori esadecimali a valori decimali, come illustrato nel seguente codice ActionScript:

```
// Unicode 0068 == "h"
var letter:Number = Number(new Number(0x0068).toString(10));
trace(String.fromCharCode(letter)); // h
```

È possibile esaminare i caratteri in varie posizioni di una stringa, come nell'esempio seguente.

Per spostarsi lungo una stringa:

1. Creare un nuovo documento Flash.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var myStr:String = "hello world!";
for (var i:Number = 0; i < myStr.length; i++) {
    trace(myStr.charAt(i));
}
```
3. Selezionare Controllo > Prova filmato per visualizzare in anteprima il documento Flash. Nel pannello Output si dovrebbe vedere lo scorrimento in ogni carattere su una riga separata.
4. Modificare il codice ActionScript in modo che esegua il tracciamento del valore ASCII per tutti i caratteri:

```
var myStr:String = "hello world!";
for (var i:Number = 0; i < myStr.length; i++) {
    trace(myStr.charAt(i) + " - ASCII=" + myStr.charCodeAt(i));
}
```

5. Salvare il documento Flash e selezionare Controllo > Prova filmato visualizzare in anteprima il file SWF.

Quando si esegue questo codice, nel pannello Output viene visualizzato quanto segue:

```
h - ASCII=104
e - ASCII=101
l - ASCII=108
l - ASCII=108
o - ASCII=111
    - ASCII=32
w - ASCII=119
o - ASCII=111
r - ASCII=114
l - ASCII=108
d - ASCII=100
! - ASCII=33
```

SUGGERIMENTO

Una stringa può essere suddivisa in un array di caratteri utilizzando il metodo `String.split()` e inserendo come delimitatore una stringa vuota (""); ad esempio, `var charArray:Array = myStr.split("")`;

È possibile utilizzare gli operatori per confrontare le stringhe. Per informazioni sull'uso di operatori con le stringhe, vedere ["Informazioni sull'uso degli operatori con le stringhe"](#) [a pagina 149](#).

Questi operatori possono essere utilizzati con istruzioni condizionali, quali `if` e `while`. Nell'esempio seguente vengono utilizzati operatori e stringhe per effettuare un confronto.

Per confrontare due stringhe:

1. Creare un nuovo documento Flash e salvarlo come **comparestr.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2) {
    trace("Uppercase letters sort first.");
}
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.
È possibile utilizzare gli operatori di uguaglianza (`==`) e di disuguaglianza (`!=`) per confrontare le stringhe con altri tipi di oggetti, come illustrato nel seguente esempio.

Per confrontare stringhe con altri tipi di dati:

1. Creare un nuovo documento Flash e salvarlo come **comparenum.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var myStr:String = "4";
var total:Number = 4;
if (myStr == total) {
    trace("Types are converted.");
}
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.
Quando si confrontano tipi di dati diversi, ad esempio stringhe e numeri, Flash cerca di convertire i tipi di dati in modo da poter effettuare un confronto.

È possibile utilizzare gli operatori di uguaglianza rigorosa (`==`) e di disuguaglianza rigorosa (`!=`) per verificare che entrambi gli oggetti di confronto siano dello stesso tipo. Il seguente esempio utilizza operatori di uguaglianza rigorosa per garantire che Flash non cerchi di convertire i tipi di dati durante il confronto dei valori.

Per impostare il confronto rigoroso di tipi di dati:

1. Creare un nuovo documento Flash e salvarlo come **comparestrict.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var str1:String = "4";
var str2:String = "5";
var total:Number = 4;
if (str1 !== total) {
    trace("Types are not converted.");
}
if (str1 !== str2) {
    trace("Same type, but the strings don't match.");
}
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato.

Per ulteriori informazioni sull'uso di operatori con le stringhe, vedere [“Informazioni sull'uso degli operatori con le stringhe” a pagina 149](#).

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, strings.fla. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Strings*.

Conversione e concatenazione di stringhe

È possibile convertire molti oggetti in stringhe tramite il metodo `toString()`. Molti oggetti incorporati dispongono, a questo scopo, di un metodo `toString()`:

```
var n:Number = 0.470;
trace(typeof(n.toString())); // stringa
```

Quando si utilizza l'operatore di addizione (+) con una combinazione di istanze String e non String, non è necessario utilizzare il metodo `toString()`. Per informazioni dettagliate sulla concatenazione, vedere la seconda procedura in questa sezione.

I metodi `toLowerCase()` e `toUpperCase()` convertono i caratteri alfabetici contenuti nella stringa rispettivamente in caratteri minuscoli e maiuscoli. Nell'esempio seguente viene dimostrato come convertire una stringa da caratteri minuscoli in caratteri maiuscoli.

Per convertire una stringa da caratteri minuscoli in maiuscoli:

1. Creare un nuovo documento Flash e salvarlo come **convert.fla**.
2. Immettere il codice seguente nel fotogramma 1 della linea temporale:

```
var myStr:String = "Dr. Bob Roberts, #9.";
trace(myStr.toLowerCase()); // dr. bob roberts, #9.
trace(myStr.toUpperCase()); // DR. BOB ROBERTS, #9.
trace(myStr); // Dr. Bob Roberts, #9.
```
3. Salvare il documento Flash e selezionare Controllo > Prova filmato.

NOTA

Dopo l'esecuzione di questi metodi, la stringa di origine rimane invariata. Per trasformare la stringa di origine, utilizzare il codice seguente:

```
myStr = myStr.toUpperCase();
```

La concatenazione di stringhe comporta l'unione sequenziale di due stringhe in un'unica stringa. Ad esempio, è possibile utilizzare l'operatore di addizione (+) per concatenare due stringhe, come illustrato nell'esempio seguente.

Per concatenare due stringhe:

1. Creare un nuovo documento Flash e salvarlo come **concat.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var str1:String = "green";
var str2:String = str1 + "ish";
trace(str2); // greenish
//
var str3:String = "yellow";
str3 += "ish";
trace(str3); // yellowish
```

Il codice mostra due metodi di concatenazione di stringhe. Il primo metodo utilizza l'operatore di addizione (+) per unire la stringa `str1` alla stringa "ish". Il secondo metodo utilizza l'operatore di addizione e assegnazione (+=) per concatenare la stringa "ish" al valore corrente di `str3`.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato.

Per concatenare due stringhe è possibile, inoltre, utilizzare il metodo `concat()` illustrato nell'esempio seguente.

Per concatenare due stringhe con il metodo concat():

1. Creare un nuovo documento Flash e salvarlo come **concat2.fla**.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
trace(str4); // Bonjour from Paris
```

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Se si utilizza l'operatore di addizione (+), oppure l'operatore di addizione e assegnazione (+=), con un oggetto di tipo String e non String, ActionScript converte automaticamente l'oggetto non String in stringhe per valutare l'espressione, come illustrato nell'esempio seguente.

```
var version:String = "Flash Player ";
var rel:Number = 8;
version = version + rel;
trace(version); // Flash Player 8
```

Per imporre la valutazione aritmetica all'operatore di addizione (+), è tuttavia possibile utilizzare le parentesi, come illustrato nel codice ActionScript seguente:

```
trace("Total: $" + 4.55 + 1.46); // Totale: $4.551.46
trace("Total: $" + (4.55 + 1.46)); // Totale: $6.01
```

È possibile utilizzare il metodo `split()` per creare un array di sottostringhe di una stringa, che viene divisa in base a un carattere delimitatore. Ad esempio, è possibile segmentare una stringa delimitata da virgolette o da tabulazioni in più stringhe.

Il codice seguente dimostra, ad esempio, come dividere un array in sottostringhe utilizzando il carattere e commerciale (&) come delimitatore.

Per creare un array di sottostringhe segmentate da un delimitatore:

1. Creare un nuovo documento Flash e salvarlo come **strsplit.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/
65";
var params:Array = queryStr.split("&", 2);
trace(params); // first=joe,last=cheng
/* params is set to an array with two elements:
   params[0] == "first=joe"
   params[1] == "last=cheng"
*/
```

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

SUGGERIMENTO

Il secondo parametro del metodo `split()` definisce le dimensioni massime dell'array. Se non si desidera limitare le dimensioni dell'array creato dal metodo `split()`, è possibile omettere il secondo parametro.

SUGGERIMENTO

Il modo più semplice di analizzare una stringa di query (una stringa delimitata dai caratteri & e =) è l'utilizzo del metodo `LoadVars.decode()`, come illustrato nel seguente codice ActionScript:

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/65";
var my_lv:LoadVars = new LoadVars();
my_lv.decode(queryStr);
trace(my_lv.first); // joe
```

Per ulteriori informazioni sull'uso di operatori con le stringhe, vedere “[Informazioni sull'uso degli operatori con le stringhe](#)” a pagina 149.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, strings.fla. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Strings*.

Restituzione di sottostringhe

I metodi `substr()` e `substring()` della classe `String` sono simili. Entrambi restituiscono infatti una sottostringa di una stringa e accettano due parametri. In entrambi i metodi, il primo parametro corrisponde alla posizione del carattere iniziale di una data stringa. Nel metodo `substr()`, tuttavia, il secondo parametro corrisponde alla *lunghezza* della sottostringa da restituire, mentre nel metodo `substring()` il secondo parametro definisce la posizione del carattere alla *fine* della sottostringa (che non è incluso nella stringa restituita). Nell'esempio seguente viene illustrata la differenza tra questi due metodi:

Per trovare una sottostringa in base alla posizione del carattere:

1. Creare un nuovo documento Flash e salvarlo come `substring.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var myStr:String = "Hello from Paris, Texas!!!";
trace(myStr.substr(11,15)); // Paris, Texas!!!
trace(myStr.substring(11,15)); // Pari
```

Il primo metodo, `substr()`, restituisce una stringa della lunghezza di 15 caratteri a partire dall'undicesimo carattere. Il secondo metodo, `substring()`, restituisce una stringa della lunghezza di quattro caratteri contenuti tra l'undicesimo e il quindicesimo indice.

3. Aggiungere il codice ActionScript seguente dopo il codice creato al punto precedente:

```
trace(myStr.slice(11, 15)); // Pari
trace(myStr.slice(-3, -1)); // !
trace(myStr.slice(-3, 26)); // !!!
trace(myStr.slice(-3, myStr.length)); // !!!
trace(myStr.slice(-8, -3)); // Texas
```

Il funzionamento del metodo `slice()` è analogo al metodo `substring()`. Se come parametri si utilizzano due numeri interi non negativi, il funzionamento è esattamente uguale. Tuttavia il metodo `slice()` accetta come parametri valori interi negativi.

4. Selezionare Controllo > Prova filmato per provare il documento Flash.

NOTA

Come parametri del metodo `slice()`, è possibile combinare numeri interi non negativi e negativi.

I metodi `indexOf()` e `lastIndexOf()` possono essere utilizzati per individuare le sottostringhe corrispondenti in una stringa, come illustrato nell'esempio seguente.

Per individuare la posizione del carattere di una sottostringa corrispondente:

1. Creare un nuovo documento Flash e salvarlo come **indexof.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var myStr:String = "The moon, the stars, the sea, the land";
trace(myStr.indexOf("the")); // 10
trace(myStr.indexOf("the", 11)); // 21
```

Il primo indice della parola *the* inizia sul decimo carattere perché il metodo `indexOf()` fa distinzione tra maiuscole e minuscole; pertanto la prima istanza di *The* non viene considerata. Per indicare la posizione di indice nella stringa dalla quale iniziare la ricerca, è possibile specificare un secondo parametro per il metodo `indexOf()`: Nel codice precedente, Flash cerca il primo indice della parola *the* presente dopo l'undicesimo carattere.

3. Aggiungere il codice ActionScript seguente dopo il codice creato al punto precedente:

```
trace(myStr.lastIndexOf("the")); // 30
trace(myStr.lastIndexOf("the", 29)); // 21
```

Il metodo `lastIndexOf()` trova l'ultima occorrenza di una sottostringa nella stringa: Invece di cercare un carattere o una sottostringa dall'inizio di una stringa, ad esempio, `lastIndexOf()` parte dalla fine di una stringa e si sposta indietro. Analogamente al metodo `indexOf()`, se con il metodo `lastIndexOf()` si include un secondo parametro, la ricerca viene effettuata da tale posizione di indice, sebbene con `lastIndexOf()` la ricerca nella stringa sia effettuata all'indietro (da destra verso sinistra).

4. Selezionare Controllo > Prova filmato per provare il documento Flash.

SUGGERIMENTO

I metodi `indexOf()` e `lastIndexOf()` fanno distinzione tra maiuscole e minuscole.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, strings.fla. Il file illustra la realizzazione di un semplice elaboratore di testi che confronta e recupera selezioni di stringa e di sottostringa.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Strings*.

Animazione, filtri e disegni

In questo capitolo viene descritto come aggiungere animazioni alle applicazioni Macromedia Flash Basic 8 e Macromedia Flash Professional 8 mediante ActionScript piuttosto che (o in concomitanza con) le animazioni basate sulla linea temporale con interpolazione di movimento o forma. L'utilizzo del codice per creare animazioni consente, il più delle volte, di ridurre la dimensione del file dell'applicazione finale e di ottimizzare le prestazioni e la coerenza dell'animazione stessa. Nel contempo, le animazioni basate sul codice ActionScript diminuiscono persino il carico di lavoro: il codice consente una scrittura più rapida ed è facile da applicare a più istanze allo stesso tempo o da riutilizzare in altre applicazioni. In questo capitolo viene descritto anche come animare mediante le nozioni di base di ActionScript, le classi Tween e TransitionManager, l'API di disegno, le classi di filtri e i metodi di fusione.

È possibile aggiungere animazione e disegnare mediante l'API di disegno, costituito dai metodi di disegno nella classe MovieClip. Questi metodi consentono di utilizzare codice per creare linee, riempimenti e forme, evitando così il ricorso agli strumenti di disegno inclusi nello strumento di creazione.

I filtri e gli altri effetti espressivi risultano fondamentali in molte applicazioni Flash, in quanto consentono di applicare un effetto e di animarlo in modo rapido. È possibile utilizzare codice per aggiungere e animare effetti di filtro, metodi di fusione e immagini bitmap.

Questo capitolo è formato dalle sezioni seguenti, nelle quali viene descritto come creare animazioni e aggiungere effetti mediante ActionScript e disegnare in ActionScript mediante l'API di disegno:

Creazione di script per animazione con ActionScript 2.0	514
Informazioni sulla memorizzazione delle bitmap nella cache, sullo scorrimento e sulle prestazioni	526
Informazioni sulle classi Tween e TransitionManager	527
Uso degli effetti filtro	545
Operazioni con i filtri mediante ActionScript	553
Gestione degli effetti filtro mediante il codice	579
Creazione di bitmap mediante la classe BitmapData	584

Informazioni sui metodi di fusione.....	587
Informazioni sull'ordine delle operazioni	590
Disegno con ActionScript	590
Nozioni fondamentali sulla modifica in scala e sulle guide porzione.....	606

Creazione di script per animazione con ActionScript 2.0

È possibile aggiungere animazioni alle applicazioni Flash mediante ActionScript 2.0 invece di utilizzare interpolazioni di movimento o di forma su una linea temporale. Nelle sezioni seguenti viene descritto come utilizzare codice per animare istanze, tra cui la modifica della trasparenza e dell'aspetto dell'istanza e lo spostamento dell'istanza intorno allo stage.

Per informazioni sull'utilizzo delle classi Tween e TransitionManager per automatizzare ulteriormente le animazioni basate sul codice, consultare “[Classe TransitionManager](#)” a pagina 1277 e “[Classe Tween](#)” a pagina 1355. Queste classi facilitano l'aggiunta di equazioni di andamento e animazioni di transizione avanzate nelle istanze di clip filmato presenti nell'applicazione. È difficile ricreare buona parte di questi effetti con ActionScript senza le classi predefinite specificate. Per ottenere l'effetto, infatti, il codice necessario comporta la scrittura di equazioni matematiche complesse .

Per ulteriori informazioni sull'animazione dei disegni creati con codice, consultare “[Disegno con ActionScript](#)” a pagina 590.

Nelle sezioni seguenti, viene descritto come creare animazioni con script:

- “[Informazioni su animazione e frequenza fotogrammi](#)” a pagina 515
- “[Applicazione della dissolvenza agli oggetti con codice](#)” a pagina 516
- “[Aggiunta di effetti di colore e luminosità con codice](#)” a pagina 518
- “[Spostamento degli oggetti mediante il codice](#)” a pagina 522
- “[Panoramica di un'immagine con codice](#)” a pagina 524

È presente un esempio di animazione con script in Flash in un file di origine di esempio, animation.fla., contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Animation*.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file di immagine in un file SWF contenente animazione con script. È possibile trovare i file di origine di esempio, *gallery_tree.fla* e *gallery_tween.fla*, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries*.

Informazioni su animazione e frequenza fotogrammi

Quando si aggiunge animazione a un'applicazione, bisogna prendere in considerazione la frequenza fotogrammi applicata al file FLA. Si tratta di un aspetto fondamentale quando si lavora con le animazioni, poiché influisce sulle prestazioni del file SWF e del computer che lo riproduce. L'impostazione di una frequenza fotogrammi troppo alta può causare problemi al processore, soprattutto quando si utilizzano molte risorse o ActionScript per creare animazioni.

È necessario prendere in considerazione l'impostazione della frequenza fotogrammi anche perché influenza sulla corretta esecuzione dell'animazione. Ad esempio, un'animazione impostata su 12 fotogrammi al secondo (fps) nella finestra di ispezione Proprietà esegue 12 fotogrammi al secondo. Se la frequenza fotogrammi del documento è impostata su 24 fps, l'animazione viene visualizzata in maniera più fluida rispetto a 12 fps. Tuttavia, l'animazione a 24 fps viene eseguita in modo più rapido rispetto a 12 fps. In questo modo, la durata complessiva (in secondi) è più breve. Pertanto, se si desidera creare un'animazione di 5 secondi con una frequenza fotogrammi più elevata, è necessario aggiungere ulteriori fotogrammi per riempire i 5 secondi (in questo modo, aumenta la dimensione totale del file dell'animazione). In genere, la dimensione del file di un'animazione di 5 secondi a 24 fps è più grande di quella di un'animazione di 5 secondi a 12 fps.

NOTA

Quando si utilizza un gestore di eventi `onEnterFrame` per creare animazioni con script, l'animazione viene eseguita alla frequenza fotogrammi del documento, in modo simile alla creazione di un'interpolazione di movimento su una linea temporale. Un'alternativa al gestore di eventi `onEnterFrame` è `setInterval` (vedere funzione `setInterval` nella Guida di riferimento ActionScript 2.0). Invece di basarsi sulla frequenza fotogrammi, le funzioni vengono chiamate a un intervallo specificato. Come `onEnterFrame`, più si utilizza `setInterval` per chiamare una funzione, più l'animazione richiede risorse del processore.

Utilizzare la frequenza fotogrammi più bassa possibile in modo da visualizzare l'animazione in modo corretto in fase di runtime. In questo modo, si riduce il carico sul processore dell'utente finale. Evitare di utilizzare una frequenza fotogrammi da 30 a 40 fps. Frequenze fotogrammi elevate riducono le prestazioni del processore. Evitare di modificare l'aspetto dell'animazione in fase di runtime.

Inoltre, soprattutto quando si realizza un'animazione basata su linea temporale, selezionare quanto prima una frequenza fotogrammi durante il processo di sviluppo. Quando si prova il file SWF, verificare la durata e la dimensione del file SWF dell'animazione. La frequenza fotogrammi influisce notevolmente sulla velocità dell'animazione.

Applicazione della dissolvenza agli oggetti con codice

Quando si lavora con i clip filmato sullo stage, è possibile scegliere di applicare la dissolvenza in entrata o in uscita a un clip filmato invece di attivare e disattivare la proprietà `_visible`. Nella procedura seguente, viene descritto come utilizzare un gestore di eventi `onEnterFrame` per animare un clip filmato.

Per dissolvere un clip filmato mediante codice:

1. Creare un nuovo documento Flash denominato `fade1.fla`.
2. Disegnare degli elementi grafici sullo stage mediante lo strumento di disegno o importare un'immagine nello stage (File > Importa > Importa nello stage).
3. Selezionare il contenuto dello stage e scegliere Elabora > Converti in simbolo.
4. Selezionare l'opzione Clip filmato e fare clic su OK per creare il simbolo.
5. Selezionare l'istanza di clip filmato nello stage e digitare `img1_mc` nella casella di testo Nome istanza nella finestra di ispezione Proprietà.
6. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
img1_mc.onEnterFrame = function() {  
    img1_mc._alpha -= 5;  
    if (img1_mc._alpha <= 0) {  
        img1_mc._visible = false;  
        delete img1_mc.onEnterFrame;  
    }  
};
```

Questo codice utilizza un gestore di eventi `onEnterFrame` chiamato ripetutamente in base alla frequenza fotogrammi del file SWF. Il numero di volte al secondo in cui il gestore di eventi viene chiamato dipende dalla frequenza fotogrammi impostata per il documento Flash. Se la frequenza è pari a 12 fotogrammi al secondo (fps), il gestore di eventi `onEnterFrame` viene chiamato 12 volte al secondo. Allo stesso modo, se la frequenza di fotogrammi del documento Flash è pari a 30 fps, il gestore di eventi viene chiamato 30 volte al secondo.

7. Selezionare Controllo > Prova filmato per provare il documento.

Il clip filmato aggiunto viene dissolto lentamente in uscita.

È possibile modificare la proprietà `_alpha` mediante la funzione `setInterval()` piuttosto che mediante un gestore di eventi `onEnterFrame` come illustrato nella procedura seguente.

Per dissolvere un oggetto mediante la funzione `setInterval()`:

1. Creare un nuovo documento Flash denominato `fade2.fla`.
2. Disegnare elementi grafici sullo stage o importare un'immagine nello stage (File > Importa > Importa nello stage).
3. Selezionare il contenuto dello stage e scegliere Elabora > Converti in simbolo.
4. Selezionare l'opzione Clip filmato e fare clic su OK per creare il simbolo.
5. Selezionare l'istanza di clip filmato nello stage e digitare `img1_mc` nella casella di testo Nome istanza nella finestra di ispezione Proprietà.
6. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
var alpha_interval:Number = setInterval(fadeImage, 50, img1_mc);
function fadeImage(target_mc:MovieClip):Void {
    target_mc._alpha -= 5;
    if (target_mc._alpha <= 0) {
        target_mc._visible = false;
        clearInterval(alpha_interval);
    }
}
```

La funzione `setInterval()` si comporta in modo leggermente diverso dal gestore di eventi `onEnterFrame`, in quanto `setInterval()` indica a Flash la frequenza con cui è necessario chiamare una determinata funzione. In questo esempio, la funzione `fadeImage()` definita dall'utente viene chiamata ogni 50 millisecondi (20 volte al secondo). La funzione `fadeImage()` decrementa il valore della proprietà `_alpha` del clip filmato corrente. Quando il valore `_alpha` è pari o inferiore a 0, l'intervallo viene cancellato. In questo modo, l'esecuzione della funzione `fadeImage()` viene interrotta.

7. Selezionare Controllo > Prova filmato per provare il documento.

Il clip filmato aggiunto viene dissolto lentamente in uscita.

Per ulteriori informazioni sulle funzioni definite dall'utente, consultare “[Definizione di funzioni globali e associate alla linea temporale](#)” a pagina 180. Per ulteriori informazioni sul gestore di eventi `onEnterFrame`, consultare `onEnterFrame` (gestore `MovieClip.onEnterFrame`) nella *Guida di riferimento ActionScript 2.0*. Per ulteriori informazioni sulla funzione `setInterval()`, consultare funzione `setInterval` nella *Guida di riferimento di ActionScript*.

È presente un esempio di animazione con script in Flash in un file di origine di esempio, `animation.fla`., contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Animation*.

Aggiunta di effetti di colore e luminosità con codice

Oltre all'utilizzo di ActionScript per impostare e animare dissolvenze alfa (vedere “[Applicazione della dissolvenza agli oggetti con codice](#)” a pagina 516), è possibile animare diversi colori ed effetti di luminosità con il codice invece di utilizzare il pannello Filtri nella finestra di ispezione Proprietà.

La procedura seguente consente di caricare un'immagine JPEG e di applicare un filtro di trasformazione del colore che modifica i canali rosso e verde nel momento in cui il puntatore del mouse viene spostato lungo l'asse x e l'asse y.

Per modificare i canali di colore mediante ActionScript:

1. Creare un nuovo documento Flash denominato **colorTrans.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var imageClip:MovieClip = this.createEmptyMovieClip("imageClip", 1);
var clipLoader:MovieClipLoader = new MovieClipLoader();
clipLoader.loadClip("http://www.helpexamples.com/flash/images/
    image1.jpg", imageClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
    var transformer:Transform = new Transform(imageClip);
    var colorTransformer:ColorTransform = transformer.colorTransform;
    colorTransformer.redMultiplier = (_xmouse / Stage.width) * 1;
    colorTransformer.greenMultiplier = (_ymouse / Stage.height) * 1;
    transformer.colorTransform = colorTransformer;
}
Mouse.addListener(mouseListener);
```

3. Selezionare Controllo > Prova filmato per provare il documento e spostare il puntatore del mouse intorno allo stage.

Il file di immagine caricato modifica i colori durante lo spostamento del mouse.

È anche possibile utilizzare la classe ColorMatrixFilter per convertire un'immagine a colori in un'immagine in bianco e nero, come nella procedura seguente.

Per utilizzare la classe ColorMatrixFilter in modo da modificare un'immagine in un'immagine in scala di grigi:

1. Creare un nuovo documento Flash denominato **grayscale.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
    var myElements_array:Array = [0.3, 0.59, 0.11, 0, 0,
        0.3, 0.59, 0.11, 0, 0,
        0.3, 0.59, 0.11, 0, 0,
        0, 0, 0, 1, 0];
    var myColorMatrix_filter:ColorMatrixFilter = new
    ColorMatrixFilter(myElements_array);
    target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);
```

Il codice precedente ha inizio nel momento in cui si importa la classe ColorMatrixFilter e consente di creare un oggetto listener, il quale viene utilizzato con una nuova istanza MovieClipLoader creata in seguito con un altro codice. Successivamente, viene creata una nuova istanza di clip filmato con il nome di istanza `img_mc` e una nuova istanza loader di clip filmato con il nome di istanza `img_mcl`. Infine, il clip filmato di origine viene caricato nel clip filmato `img_mc` sullo stage. Quando l'immagine viene caricata, viene chiamato il gestore di eventi `onLoadInit` e associata una classe ColorMatrixFilter a essa.

3. Selezionare Controllo > Prova filmato per provare il documento.

L'immagine caricata nello stage viene modificata in un'immagine in scala di grigi.

Visualizzare l'immagine online ([http://www.helpexamples.com/flash/images/
image1.jpg](http://www.helpexamples.com/flash/images/image1.jpg)) per constatarne il colore originale.

È anche possibile impostare la luminosità di un'immagine mediante il codice ActionScript come mostrato nella procedura seguente.

Per modificare la luminosità di un'immagine:

1. Creare un nuovo documento Flash denominato **brightness.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
    var myElements_array:Array = [1, 0, 0, 0, 100,
        0, 1, 0, 0, 100,
        0, 0, 1, 0, 100,
        0, 0, 0, 1, 0];
    var myColorMatrix_filter:ColorMatrixFilter = new
    ColorMatrixFilter(myElements_array);
    target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
    img_mc);
```

Questo blocco di codice si basa sulla classe MovieClipLoader per caricare un file JPEG esterno. Quando l'immagine viene caricata, viene chiamato il gestore di eventi onLoadInit della classe MovieClipLoader e modificata la luminosità dell'immagine a 100 mediante il filtro ColorMatrixFilter.

3. Selezionare Controllo > Prova filmato per provare il documento.

Quando si prova il file SWF, viene modificata la luminosità dell'immagine caricata nel file SWF. Visualizzare l'immagine online (<http://www.helpexamples.com/flash/images/image2.jpg>) per constatarne l'aspetto originale.

È presente un esempio di animazione con script in Flash in un file di origine di esempio, **animation.fla**, contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Animation*.

È anche possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico durante il caricamento di file di immagine in un file SWF, il quale contiene animazione con script. È possibile trovare i file di origine di esempio, *gallery_tree.fla* e *gallery_tween.fla*, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries*.

Spostamento degli oggetti mediante il codice

Lo spostamento di un oggetto mediante ActionScript è molto simile alla modifica della proprietà `_alpha` di un oggetto, anche se, a differenza di questa, le proprietà `_x` o `_y` vengono modificate.

Nella procedura seguente, un'immagine JPEG caricata in modo dinamico viene animata e spostata orizzontalmente sullo stage.

Per spostare un'istanza sullo stage mediante codice:

1. Creare un nuovo documento Flash denominato **moveClip.fla**.
2. Impostare la frequenza fotogrammi del documento su 24 fps nella finestra di ispezione Proprietà.
Se si imposta una frequenza fotogrammi più alta, ad esempio 24 fps, l'animazione risulta più attenuata.
3. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
// Crea un'istanza di clip filmato.
this.createEmptyMovieClip("img1_mc", 10);
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function (target_mc:MovieClip):Void {
    target_mc._x = Stage.width;
    target_mc.onEnterFrame = function() {
        target_mc._x -= 3; // Decrementa la posizione _x corrente di 3 pixel
        if (target_mc._x <= 0) {
            target_mc._x = 0;
            delete target_mc.onEnterFrame;
        }
    };
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
```

```
// Carica un'immagine nel clip filmato.  
img_mc1.loadClip("http://www.helplexamples.com/flash/images/image1.jpg",  
    img1_mc);
```

In questo esempio, un'immagine esterna viene caricata da un server Web remoto e, a caricamento completato, viene animata orizzontalmente sullo stage. Invece di utilizzare un gestore di eventi `onEnterFrame`, è anche possibile animare l'immagine mediante la funzione `setInterval()`.

4. Selezionare Controllo > Prova filmato per provare il documento.

L'immagine, viene caricata e animata dal lato destro all'angolo superiore sinistro dello stage.

Per informazioni sull'utilizzo del gestore di eventi `onEnterFrame` o sulla funzione `setInterval()` per animare l'immagine, consultare “[Applicazione della dissolvenza agli oggetti con codice](#)” a pagina 516.

È presente un esempio di animazione con script in Flash in un file di origine di esempio, `animation.fla`, contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Animation*.

È anche possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file di immagine in un file SWF contenente animazione con script. È possibile trovare i file di origine di esempio, `gallery_tree.fla` e `gallery_tween.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Galleries*.

Panoramica di un'immagine con codice

Grazie ad ActionScript, è possibile eseguire la panoramica di immagini estese all'interno di documenti Flash in modo semplice. Questo è utile quando l'immagine non si adatta allo stage o si desidera creare un effetto di animazione in cui un clip filmato viene spostato da una parte all'altra dello stage. Ad esempio, se si dispone di un'immagine panoramica di dimensioni più estese rispetto allo stage e non si desidera ridurre l'immagine né aumentare le dimensioni dello stage, è possibile creare un clip filmato che funga da maschera per l'immagine più estesa.

Nella procedura seguente, viene descritto come mascherare un clip filmato in modo dinamico e animare un'immagine dietro la maschera mediante un gestore di eventi `onEnterFrame`.

Per eseguire la panoramica di un'istanza sullo stage mediante codice:

1. Creare un nuovo documento Flash denominato `pan fla`.
2. Impostare la frequenza fotogrammi del documento su 24 fps nella finestra di ispezione Proprietà.

Se si imposta una frequenza fotogrammi più alta, ad esempio 24 fps, l'animazione risulta più attenuata.

3. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
System.security.allowDomain("http://www.helpexamples.com/");
// Inizializzare variabili
var direction:Number = -1;
var speed:Number = 5;
// Creare il clip in cui caricare un'immagine
this.createEmptyMovieClip("img_mc", 10);
// Creare un clip da utilizzare come maschera
this.createEmptyMovieClip("mask_mc", 20);
// Disegnare/creare una maschera mediante l'API di disegno.
with (mask_mc) {
    beginFill(0xFF0000, 0);
    moveTo(0, 0);
    lineTo(300, 0);
    lineTo(300, 100);
    lineTo(0, 100);
    lineTo(0, 0);
    endFill();
}

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
    // Impostare la maschera del clip filmato di destinazione su mask_mc
    target_mc.setMask(mask_mc);
    target_mc.onEnterFrame = function() {
        target_mc._x += speed * direction;
    }
}
```

```

        // Se target_mc si trova in posizione angolata, invertire la
        direzione dell'animazione
        if ((target_mc._x <= -(target_mc._width-mask_mc._width)) ||
        (target_mc._x >= 0)) {
            direction *= -1;
        }
    };
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);

```

Nella prima sezione di codice in questo esempio di codice, vengono definite due variabili: `direction` e `speed`. La variabile `direction` controlla lo scorrimento dell'immagine mascherata, da sinistra a destra (1) o da destra a sinistra (-1). La variabile `speed` controlla il numero dei pixel che vengono spostati ogni volta che viene chiamato il gestore di eventi `onEnterFrame`. Ai numeri più alti corrisponde uno spostamento più veloce dell'animazione, anche se questa apparirà leggermente meno fluida.

La sezione di codice seguente consente di creare due clip filmato vuoti: `img_mc` e `mask_mc`. Un rettangolo di 300 x 100 pixel viene disegnato all'interno del filmato `mask_mc` mediante l'API di disegno. Quindi, viene creato un nuovo oggetto (`mcl_obj`) da usare come listener per un'istanza MovieClipLoader creata in un blocco di codice finale. Questo oggetto definisce un listener di eventi per l'evento `onLoadInit`, applica una maschera all'immagine caricata in modo dinamico e imposta l'animazione scorrevole. Quando l'immagine raggiunge il bordo destro o sinistro della maschera, l'animazione viene invertita.

L'ultimo blocco di codice definisce un'istanza MovieClipLoader, specifica l'oggetto listener indicato in precedenza e inizia a caricare l'immagine JPEG nel clip filmato `img_mc` creato precedentemente.

4. Selezionare Controllo > Prova filmato per provare il documento.

Viene caricata l'immagine, la quale si anima in avanti e all'indietro in un movimento panoramico (movimento da una parte all'altra). L'immagine viene mascherata nella fase di runtime. È possibile visualizzare l'immagine originale online (<http://www.helpexamples.com/flash/images/image1.jpg>).

Informazioni sulla memorizzazione delle bitmap nella cache, sullo scorrimento e sulle prestazioni

Flash Player 8 introduce la memorizzazione delle bitmap nella cache, la quale consente di ottimizzare le prestazioni dei clip filmato non variabili nelle applicazioni. Quando si imposta la proprietà `MovieClip.cacheAsBitmap` o `Button.cacheAsBitmap` su true, Flash Player memorizza nella cache una rappresentazione bitmap interna del clip filmato o dell'istanza del pulsante. In questo modo, è possibile migliorare le prestazioni per i clip filmato con contenuto vettoriale complesso. Tutti i dati vettoriali per un clip filmato con una bitmap memorizzata nella cache vengono disegnati sulla bitmap anziché sullo stage principale.



La bitmap viene copiata sullo stage principale sotto forma di pixel non allungati né ruotati, agganciati ai limiti di pixel più vicini. I pixel vengono mappati 1 a 1 con l'oggetto principale. Se i limiti della bitmap cambiano, questa viene ricreata anziché essere allungata.

Per ulteriori informazioni sulla memorizzazione delle bitmap nella cache e sulle istanze di clip filmato, consultare le seguenti sezioni in [Capitolo 11, “Operazioni con i clip filmato”](#):

- [“Informazioni sulla memorizzazione nella cache e sullo scorrimento dei clip filmato in ActionScript” a pagina 401](#)
- [“Memorizzazione di un clip filmato nella cache” a pagina 406](#)
- [“Impostazione dello sfondo di un clip filmato” a pagina 408](#)

L'uso della proprietà `cacheAsBitmap` è più adatto ai clip filmato che includono contenuto statico e non vengono modificati in scala e ruotati frequentemente. Con questo tipo di clip filmato, la proprietà `cacheAsBitmap` può consentire miglioramenti delle prestazioni in caso di conversione di un clip filmato (quando viene modificata la posizione di `x` e `y`). Per ulteriori informazioni su questa funzione, consultare [“Quando attivare la memorizzazione nella cache” a pagina 403](#).

È possibile trovare un file di origine di esempio in cui viene descritto come applicare la memorizzazione delle bitmap nella cache ad un'istanza. Individuare il file denominato `cacheBitmap.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/CacheBitmap*.

È anche possibile trovare un file di origine di esempio in cui viene descritto come applicare la memorizzazione delle bitmap nella cache a testo scorrevole. Individuare il file denominato flashtype.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio*\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/FlashType.

Informazioni sulle classi Tween e TransitionManager

Con Flash Basic 8 e Flash Professional 8 vengono installate anche due classi particolarmente efficaci: le classi Tween e TransitionManager. In questa sezione, viene descritto come utilizzare queste classi con i clip filmato e i componenti di Macromedia V2 (inclusi in Flash MX 2004 e Flash 8) per aggiungere animazione ai file SWF in modo semplice.

Se viene creata una presentazione a diapositive o un'applicazione form mediante Flash Professional 8 (solo ActionScript 2.0), è possibile selezionare i comportamenti che aggiungono diversi tipi di transizioni fra le diapositive, analogamente a quando si crea una presentazione PowerPoint. Questa funzionalità viene aggiunta in un'applicazione a schermo mediante le classi Tween e TransitionManager, che generano codice ActionScript inteso ad aggiungere animazione alle schermate a seconda del comportamento scelto.

Le classi Tween e TransitionManager possono anche essere usate al di fuori dei documenti basati su schermate, sia in Flash Basic 8 che in Flash Professional 8. Ad esempio, è possibile usarle con il set di componenti di versione 2 di Macromedia Component Architecture o con i clip filmato. Se si desidera modificare il modo in cui un componente ComboBox si anima, è possibile usare la classe TransitionManager per aggiungere *andamento* nel momento in cui si apre il menu. L'andamento si riferisce all'accelerazione o decelerazione graduale dell'animazione, che consentono di rendere più realistica l'animazione. È anche possibile usare le classi Tween e TransitionManager invece delle interpolazioni di movimento sulla linea temporale o il codice personalizzato, per creare il proprio sistema di menu con animazione.



Le classi Tween e TransitionManager sono disponibili soltanto in ActionScript 2.0, Flash Basic 8 e Flash Professional 8.

Per informazioni sui metodi e le proprietà della classe Tween, consultare il Capitolo 51, “Classe Tween” nella *Guida di riferimento dei componenti*. Per informazioni sui metodi e le proprietà della classe TransitionManager, consultare il Capitolo 48, “Classe TransitionManager” nella *Guida di riferimento dei componenti*. Per informazioni sulle operazioni con i pacchetti, consultare “[Operazioni con i pacchetti dei filtri](#)” a pagina 547.

È possibile trovare un file di origine di esempio che utilizza queste classi per aggiungere animazione con script. Individuare tweenProgress.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/tween ProgressBar*.

Per ulteriori informazioni sulle classi Tween e TransitionManager , consultare i seguenti argomenti:

- “[Aggiunta di interpolazioni e transizioni in un file in Flash Professional 8 \(soltanto Flash Professional 8\)](#)” a pagina 528
- “[Animazione mediante le classi Tween e TransitionManager](#)” a pagina 531
- “[Informazioni sulle classi e i metodi di andamento](#)” a pagina 534
- “[Informazioni sulla classe Tween](#)” a pagina 535
- “[Uso della classe Tween](#)” a pagina 537
- “[Uso combinato delle classi Tween e TransitionManager](#)” a pagina 543

Aggiunta di interpolazioni e transizioni in un file in Flash Professional 8 (soltanto Flash Professional 8)

NOTA

In questa sezione, viene descritto l'aspetto finale in Flash Professional di una presentazione di diapositive di Flash Professional a cui sono state aggiunte interpolazioni e transizioni. Tuttavia, è possibile aggiungere transizioni e interpolazioni alle applicazioni Flash Basic 8 (o flash Professional 8) mediante codice. Nelle sezioni seguenti, vengono presentati alcuni esempi.

Le classi Tween e TransitionManager sono progettate in modo che sia possibile aggiungere animazioni ad alcune parti del file SWF mediante codice ActionScript semplice. L'ambiente di creazione Flash contiene dei comportamenti che consentono di usare queste classi predefinite per le transizioni in un'applicazione basata sulle schermate. Per creare una presentazione basata su diapositive o un'applicazione form, è possibile selezionare comportamenti che aggiungono diversi tipi di transizioni tra le diapositive.

Prima di iniziare ad utilizzare queste transizioni con i clip filmato in Flash, è opportuno verificare il loro effetto quando si usa un'applicazione basata sulle schermate.

Per visualizzare il codice ActionScript che crea una transizione in una presentazione basata su diapositive:

1. Selezionare File > Nuovo e creare una nuova presentazione in Flash Professional 8.
2. Nella scheda Generali, selezionare Presentazione Flash, quindi fare clic su OK.
3. Selezionare Finestra > Comportamenti per aprire il pannello Comportamenti.
4. Fare clic su Aggiungi comportamento (+).
5. Selezionare Schermata > Transizione dal menu a comparsa in modo che venga visualizzata la finestra di dialogo Transizioni.
6. Selezionare la transizione Zoom.
7. Digitare 1 nella casella di testo Durata.
8. Selezionare Rimbalzo dal menu a comparsa Andamento.
9. Fare clic su OK per applicare le impostazioni e chiudere la finestra di dialogo.

In questo modo vengono aggiunte circa 15 righe di codice ActionScript direttamente sulla diapositiva. Nel frammento di codice seguente, viene mostrato il codice di transizione necessario:

```
mx.transitions.TransitionManager.start(eventObj.target,  
    {type:mx.transitions.Zoom, direction:0, duration:1,  
     easing:mx.transitions.easing.Bounce.easeOut, param1:empty,  
     param2:empty});
```

Questa riga di codice chiama la classe TransitionManager, quindi applica la transizione Zoom mediante il metodo di andamento specificato

`mx.transitions.easing.Bounce.easeOut`. In questo caso, la transizione viene applicata alla diapositiva selezionata. Se si desidera applicare questo effetto a un clip filmato, è possibile modificare il codice ActionScript da usare nelle animazioni Flash. Modificare il codice associato al simbolo di un clip filmato è semplice: modificare il primo parametro da `eventObj.target` nel nome di istanza del clip filmato desiderato.

Flash viene fornito con dieci transizioni, che possono essere personalizzate mediante i metodi di andamento e diversi parametri opzionali. Ricordare che l'andamento si riferisce all'accelerazione o decelerazione graduale dell'animazione, che consentono di rendere più realistica l'animazione. Ad esempio, il movimento di una palla può gradualmente accelerare subito dopo l'inizio dell'animazione e decelerare prima di fermarsi completamente alla fine dell'animazione. Esistono molte equazioni per questa accelerazione e decelerazione, che modificano l'animazione dell'andamento.

Nella tabella seguente, vengono descritte le transizioni comprese in Flash Basic 8 (mediante codice) e Flash Professional 8 (mediante codice o comportamenti):

Transizione	Descrizione
Diaframma	Rivela la schermata o il clip filmato mediante la maschera animata di una forma che viene ingrandita.
A comparsa	Rivela la schermata o il clip filmato mediante la maschera animata di una forma che si sposta orizzontalmente.
Dissolvenza pixel	Maschera la schermata o il clip filmato mediante rettangoli che appaiono e scompaiono.
Tendine	Rivela la schermata o il clip filmato successivi mediante rettangoli che appaiono e scompaiono.
Dissolvenza	Applica alla schermata o al clip filmato la dissolvenza in entrata o in uscita.
A entrata	Scorre nella schermata o nel clip filmato da una direzione specifica.
Zoom	Ingrandisce o riduce la schermata o il clip filmato.
Comprimi	Modifica in scala orizzontalmente o verticalmente la schermata o il clip filmato.
Ruota	Ruota la schermata o il clip filmato.
Foto	Fa in modo che la schermata o il clip filmato appaiano come in una fotografia.

A ogni transizione possono essere applicate delle funzioni personalizzate lievemente diverse. La finestra di dialogo Transizioni consente di visualizzare un'animazione di esempio prima di applicare il suo effetto alla diapositiva o form.

SUGGERIMENTO

Per visualizzare un'anteprima del funzionamento di ogni singola transizione con i diversi metodi delle classi di andamento, è possibile fare doppio clic su Transition.swf in *unità di avvio\Programmi\Macromedia\Flash 8\lingua\First Run\Behaviors* oppure *Macintosh HD:Applications:Macromedia Flash 8:First Run:Behaviors*: per visualizzare il file SWF nel lettore autonomo.

Animazione mediante le classi Tween e TransitionManager

Le classi Tween e TransitionManager possono essere usate in Flash Basic 8 e Flash Professional 8 per aggiungere animazione a clip filmato, componenti e fotogrammi mediante ActionScript. Se non si usano queste classi, è necessario scrivere il codice personalizzato che consenta di animare i clip filmato o modificare il loro livello di trasparenza (alfa) e le coordinate (posizione). Se si aggiunge andamento all'animazione, il codice ActionScript e le relative equazioni matematiche possono complicarsi parecchio. Tuttavia, se si desidera modificare l'andamento su una particolare animazione e usare queste classi predefinite, è possibile selezionare una classe diversa anziché cercare di capire quali equazioni matematiche sono necessarie per creare un'animazione fluida.

La procedura seguente consente di animare un clip filmato in modo che venga ingrandito sullo stage mediante la classe TransitionManager.

Per animare un clip filmato mediante la classe TransitionManager:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Fare clic su OK per creare il nuovo file FLA.
3. Salvare il file FLA come **zoom.fla**.
4. Selezionare File > Importa > Importa nello stage e selezionare nel disco rigido un'immagine da importare nel file FLA.

L'immagine viene importata nel file sotto forma di immagine bitmap, quindi occorre convertirla manualmente in simbolo di clip filmato.

5. Fare clic su Apri per importare l'immagine.
6. Selezionare l'immagine impostata nello stage e scegliere Elabora > Converti in simbolo.
7. Denominare il simbolo **img1** e accertarsi di impostare il comportamento su Movie Clip. Per impostazione predefinita, il punto di registrazione del simbolo si trova nell'angolo superiore sinistro dello stesso.
8. Fare clic su OK per convertire l'immagine bitmap in clip filmato.
9. Sempre con l'immagine selezionata, aprire la finestra di ispezione Proprietà (Finestra > Proprietà> Proprietà) e assegnare al clip filmato il nome di istanza **img1_mc**.

- 10.** Selezionare il fotogramma 1 della linea temporale principale e, nel pannello Azioni, aggiungere il codice ActionScript seguente:

```
mx.transitions.TransitionManager.start(img1_mc,  
    {type:mx.transitions.Zoom, direction:0, duration:1,  
     easing:mx.transitions.easing.Bounce.easeOut, param1:empty,  
     param2:empty});
```



Per informazioni sulle operazioni con i pacchetti, consultare “[Operazioni con i pacchetti dei filtri](#)” a pagina 547.

- 11.** Selezionare Controllo > Prova filmato per provare l'animazione.

L'immagine viene ingrandita e presenta un effetto di leggero rimbalzo prima di tornare alle dimensioni originali. Se l'animazione si muove troppo velocemente, aumentarne la durata (nel frammento di codice precedente) da un secondo a due/tre secondi (ad esempio, duration:3).

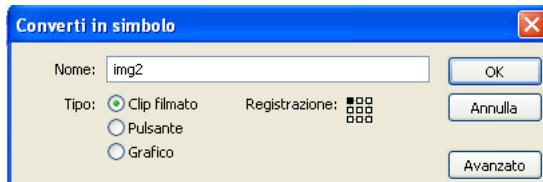
Potrebbe accadere che l'immagine venga ancorata nell'angolo superiore sinistro e si ingrandisca verso l'angolo inferiore destro. Questo effetto è diverso dall'anteprima visualizzata nella finestra di dialogo Transizione.

La creazione di animazioni complesse è una procedura semplice se si usano le classi Tween e TransitionManager e non richiede la creazione di interpolazioni di forma o movimento sulla linea temporale. Inoltre, fatto ancora più importante, non è necessario scrivere complicate equazioni matematiche per creare i metodi di andamento. Se si desidera che le immagini vengano ingrandite dal centro invece che essere ancorate a un angolo, modificare il punto di registrazione del simbolo quando si converte l'immagine da bitmap a simbolo.

Per ingrandire le immagini dal centro:

1. Completare i passaggi della procedura precedente.
2. Aprire zoom.fla e selezionare File > Salva con nome per salvare una nuova copia del documento.
Salvare il file come **zoom2.fla**.
3. Trascinare una copia del simbolo di bitmap dal pannello Libreria allo stage, di fianco al simbolo di clip filmato corrente.
4. Con l'immagine bitmap selezionata sullo stage, premere F8 per convertire il simbolo in clip filmato.
Denominare il simbolo **img2**.

5. Nella finestra di dialogo Converti in simbolo, fare clic sul punto centrale della griglia 3x3 per impostare il punto di registrazione sul centro della bitmap, quindi fare clic su OK.



6. Selezionare il nuovo clip filmato sullo stage e assegnare un nome di istanza **img2_mc** mediante la finestra di ispezione Proprietà.
7. Selezionare il fotogramma 1 della linea temporale principale e aggiungere il codice ActionScript seguente:

```
mx.transitions.TransitionManager.start(img2_mc,  
{type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,  
duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

8. Selezionare Controllo > Prova filmato per provare l'animazione.

Il secondo clip filmato viene ingrandito dal centro del simbolo invece che dall'angolo.

NOTA

Alcune transizioni sono particolarmente sensibili al posizionamento del punto di registrazione e la sua modifica può incidere in modo molto significativo sull'aspetto dell'animazione in un file SWF. Ad esempio, se il punto di registrazione si trova nell'angolo superiore sinistro quando si utilizza la transizione Zoom, la transizione viene avviata da quella posizione.

Per informazioni sui metodi e le proprietà della classe Tween, consultare il Capitolo 51, “Classe Tween” nella *Guida di riferimento dei componenti*. Per informazioni sui metodi e le proprietà della classe TransitionManager, consultare il Capitolo 48, “Classe TransitionManager” nella *Guida di riferimento dei componenti*.

È possibile trovare un file di origine di esempio che utilizza queste classi per aggiungere animazione con script. Individuare tweenProgress.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar*.

Informazioni sulle classi e i metodi di andamento

Nella sezione “[Aggiunta di interpolazioni e transizioni in un file in Flash Professional 8 \(soltanto Flash Professional 8\)](#)” a pagina 528, viene descritto come utilizzare la classe di andamento Rimbalzo per aggiungere un effetto di rimbalzo al clip filmato. Oltre a Rimbalzo, Flash 8 dispone di cinque classi di andamento supplementari, descritte nella tabella seguente:

Transizione	Descrizione
Indietro	Estende l'animazione oltre l'intervallo di transizione a una o entrambe le estremità per fornire un leggero effetto di "sconfinamento".
Rimbalzo	Applica un effetto di rimbalzo 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.
Elastico	Aggiunge un effetto elastico che fuoriesce dall'intervallo di transizione a una o entrambe le estremità. La quantità di elasticità non dipende dalla durata.
Normale	Applica un movimento rallentato a una o entrambe le estremità. Questa funzione consente di aggiungere un effetto di accelerazione, di rallentamento o entrambi.
Forte	Applica un movimento rallentato a una o entrambe le estremità. Questo effetto è simile a quello fornito da Normale, ma più pronunciato.
Nessuno	Applica un movimento uguale dall'inizio alla fine, senza effetti, rallentamento o accelerazione. Questa transizione è nota anche come transizione lineare.

Ciascuna di queste sei classi di andamento presenta tre metodi di andamento, descritti nella tabella seguente:

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

Per aprire queste classi in Flash o nell'Editor di ActionScript editor, accedere a *disco rigido\Programmi\Macromedia\Flash 8\lingua\First Run\Classes\mx\transitions\easing* in Windows (installazione predefinita) o *Macintosh HD/Applications/Macromedia Flash 8/First Run/Classes/mx/transitions/easing*.

La procedura relativa all'ingrandimento delle immagini inclusa in “[Animazione mediante le classi Tween e TransitionManager](#)” a pagina 531 si basa sulla classe di andamento e metodo mx.transitions.easing.Bounce.easeOut. Nella cartella sul disco rigido, ActionScript si riferisce al metodo `easeOut()` compreso nella classe Bounce.as. Questo file di ActionScript costituisce la cartella degli andamenti.

Per informazioni sui metodi e le proprietà della classe Tween, consultare il Capitolo 51, “Classe Tween” nella *Guida di riferimento dei componenti*. Per informazioni sui metodi e le proprietà della classe TransitionManager, consultare il Capitolo 48, “Classe TransitionManager” nella *Guida di riferimento dei componenti*.

SUGGERIMENTO

Per visualizzare un'anteprima del funzionamento di ogni singola transizione con i diversi metodi delle classi di andamento, è possibile fare doppio clic su Transition.swf in *unità di avvio\Programmi\Macromedia\Flash 8\lingua\First Run\Behaviors* oppure *Macintosh HD:Applications:Macromedia Flash 8:First Run:Behaviors*: per visualizzare il file SWF nel lettore autonomo.

È possibile trovare un file di origine di esempio che utilizza queste classi per aggiungere animazione con script. Individuare tweenProgress.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar*.

Informazioni sulla classe Tween

La classe Tween consente di spostare, ridimensionare e applicare la dissolvenza ai clip filmato sullo stage. La funzione di costruzione per la classe mx.transitions.Tween presenta i nomi e i tipi di parametro seguenti:

```
function Tween(obj, prop, func, begin, finish, duration, useSeconds) {  
    // codice ...  
}
```

obj è il clip filmato che l'istanza Tween considera come oggetto target.

prop è un nome della proprietà in **obj** in cui è possibile interpolare i valori.

func è il metodo di andamento che consente di calcolare un effetto di andamento dei valori delle proprietà dell'oggetto cui applicare l'interpolazione di movimento.

begin è un numero che indica il valore iniziale di **prop** (la proprietà dell'oggetto target cui applicare l'interpolazione di movimento).

`finish` è un numero che indica il valore finale di `prop` (la proprietà dell'oggetto `target` cui applicare l'interpolazione di movimento).

`duration` è un numero che indica la lunghezza temporale dell'interpolazione di movimento. Se omesso, per impostazione predefinita la durata viene impostata come infinita.

`useSeconds` è un valore booleano associato al valore da specificare nel parametro `duration`, nel quale viene specificato di utilizzare secondi se è `true` o fotogrammi se è `false`.

Ad esempio, si supponga di voler spostare un clip filmato all'interno dello stage. È possibile aggiungere fotogrammi chiave alla linea temporale e inserire tra loro un'interpolazione di forma o movimento, scrivere del codice in un gestore di eventi `onEnterFrame`, oppure usare `setInterval()` per chiamare una funzione a intervalli regolari. Se si usa la classe `Tween`, si dispone di un'altra opzione che consente di modificare le proprietà `_x` e `_y` di un clip filmato. Inoltre, è possibile aggiungere i metodi di andamento descritti in precedenza. Per sfruttare al meglio i vantaggi della classe `Tween`, usare il codice ActionScript seguente:

```
new mx.transitions.Tween(ball_mc, "_x",
    mx.transitions.easing.Elastic.easeOut, 0, 300, 3, true);
```

Questo frammento di codice ActionScript crea una nuova istanza della classe `Tween`, che anima il clip filmato `ball_mc` sullo stage lungo l'asse `x` (da sinistra a destra). Il clip filmato si anima da 0 a 300 pixel in tre secondi e ActionScript applica un metodo di andamento elastico. Questo significa che la palla si estende oltre i 300 pixel sull'asse `x` prima di animarsi all'indietro mediante un effetto di movimento fluido.

È possibile trovare un file di origine di esempio che utilizza queste classi per aggiungere animazione con script. Individuare `tweenProgress.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar*.

Uso della classe Tween

Se la classe Tween viene usata in più punti del documento Flash, è possibile scegliere di usare un'istruzione `import` che consente di importare la classe Tween e i metodi di andamento anziché indicare i nomi completi della classe ogni volta che vengono usati, come illustrato nella procedura seguente:

Per importare e utilizzare la classe Tween:

1. Creare un nuovo documento e denominarlo `easeTween.fla`.
2. Creare un clip filmato sullo stage.
3. Selezionare l'istanza di clip filmato e digitare `ball_mc` nella casella di testo Nome istanza della finestra di ispezione Proprietà.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_x", Elastic.easeOut, Stage.width, 0, 3, true);
```

L'esempio di codice si basa su due istruzioni `import`. La prima istruzione importa solo la classe `mx.transitions.Tween` e la seconda istruzione `import` si basa sul carattere `jolly (*)` per importare tutte le sei classi di andamento mediante un'unica stringa di codice. La seconda istruzione consente di importare un pacchetto di classi completo.



Per informazioni sulle operazioni con i pacchetti, consultare “[Operazioni con i pacchetti dei filtri](#)” a pagina 547.

5. Selezionare Controllo > Prova filmato per visualizzare l'animazione.

Nella documentazione di Flash, il *pacchetto* è costituito da directory contenenti uno o più file di classe. Risiede in una directory del percorso di classe definita. In questo caso, il pacchetto è ubicato nella cartella `C:\Program Files\Macromedia\Flash 8\lingua\First Run\Classes\mx\transitions\.easing` (Windows) oppure `HD:Applications:Macromedia Flash 8:First Run:Classes:mx:transitions:easing` (Macintosh). È abbastanza chiaro che la possibilità di importare un pacchetto completo è molto più vantaggiosa che dovere importare le sei classi separatamente. Anziché fare riferimento alla classe `mx.transitions.Tween`, ActionScript fa riferimento direttamente alla classe `Tween`; allo stesso modo, invece di usare i nomi completi per le classi di andamento, ad esempio `mx.transitions.easing.Elastic.easeOut`, è possibile digitare `Elastic.easeOut` nel codice ActionScript. Per ulteriori informazioni, consultare “[Operazioni con i pacchetti dei filtri](#)” a pagina 547.

Mediante codice simile a questo, è possibile impostare la proprietà `_alpha` per dissolvere le istanze in entrata e in uscita, invece di `_x`, come illustrato nella procedura seguente.

Per dissolvere istanze mediante la classe Tween:

1. Creare un nuovo documento e denominarlo **fadeTween.fla**.
2. Creare un clip filmato sullo stage.
3. Selezionare l'istanza di clip filmato e digitare **ball_mc** nella casella di testo Nome istanza della finestra di ispezione Proprietà.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 3, true);
```

Anziché spostarsi all'interno dello stage, **ball_mc** ora passa da visibile al 100% a completamente trasparente nell'arco di tre secondi. Per dissolvere il simbolo in uscita in modo più rapido, modificare il parametro di durata da 3 a 1 o 2.

5. Selezionare Controllo > Prova filmato per visualizzare l'animazione.

Se si modifica la frequenza fotogrammi del documento, l'animazione viene visualizzata in maniera più fluida durante l'esecuzione. Per informazioni sull'animazione e la frequenza fotogrammi, consultare “[Informazioni su animazione e frequenza fotogrammi](#)” [a pagina 515](#).

Anziché usare i secondi, è anche possibile applicare la dissolvenza al simbolo nell'arco di alcuni fotogrammi. Per impostare la durata in numero di fotogrammi anziché in secondi, modificare il parametro finale della classe Tween, **useSeconds**, da **true** a **false**. L'impostazione del parametro su **true** indica a Flash che la durata specificata è espressa in secondi. Se il parametro viene impostato su **false**, la durata specificata rappresenta invece il numero di fotogrammi che si desidera utilizzare per l'interpolazione. Nella procedura seguente, viene descritto come impostare un'interpolazione in fotogrammi anziché in secondi.

Per impostare la durata in fotogrammi anziché in secondi:

1. Creare un nuovo documento e denominarlo **framesTween.fla**.
2. Creare un clip filmato sullo stage.
3. Selezionare l'istanza di clip filmato e digitare **ball_mc** nella casella di testo Nome istanza della finestra di ispezione Proprietà.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 24, false);
```

Questo codice dissolve in uscita ball_mc mediante il metodo di andamento Strong.easeIn. Anziché dissolvere l'istanza in tre secondi, la dissolve nell'arco di 24 fotogrammi.

5. Selezionare Controllo > Prova filmato per visualizzare l'animazione.

Dopo qualche istante, l'istanza si dissolve in uscita nell'arco di 24 fotogrammi.

6. Tornare all'ambiente di creazione e aprire la finestra di ispezione Proprietà.

7. Impostare la frequenza fotogrammi del documento su 24 fps.

Se si incrementa la frequenza fotogrammi del file FLA, l'istanza viene dissolta prima. Per informazioni sull'animazione e la frequenza fotogrammi, consultare “[Informazioni su animazione e frequenza fotogrammi](#)” a pagina 515.

L'uso dei fotogrammi al posto dei secondi offre una maggiore flessibilità; tuttavia, è importante ricordare che la durata è relativa alla frequenza di fotogrammi del documento Flash corrente. Se il documento usa una frequenza di 12 fotogrammi al secondo (f/s), il frammento di codice precedente dissolve l'istanza in due secondi (24 fotogrammi a 12 f/s = 2 secondi). Tuttavia, se la frequenza è pari a 24 fotogrammi al secondo, lo stesso codice dissolve l'istanza in un secondo (24 fotogrammi a 24 f/s = 1 secondo). Se si usano i fotogrammi per misurare la durata, è possibile modificare notevolmente la velocità dell'animazione modificando la frequenza di fotogrammi del documento senza alterare il codice ActionScript. La classe Tween offre diverse altre utili funzioni; ad esempio, è possibile creare un gestore di eventi che viene eseguito dopo il completamento dell'animazione, come illustrato nella procedura seguente.

Per attivare codice quando un'animazione è completata:

1. Creare un nuovo documento e denominarlo triggerTween.fla.

2. Creare un clip filmato sullo stage.

3. Selezionare l'istanza di clip filmato e digitare ball_mc nella casella di testo Nome istanza della finestra di ispezione Proprietà.

4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var tween_handler:Object = new Tween(ball_mc, "_alpha", Strong.easeIn,
    100, 0, 3, true);
tween_handler.onMotionFinished = function() {
    trace("onMotionFinished triggered");
};
```

Se si prova questo codice ActionScript nel file FLA, il messaggio di attivazione di `onMotionFinished` viene visualizzato nel pannello Output al termine della dissolvenza di `ball_mc` sullo stage.

5. Selezionare Controllo > Prova filmato per visualizzare l'animazione.

Dopo qualche istante, l'istanza viene dissolta in uscita. Al termine dell'interpolazione, viene visualizzato un messaggio nel pannello Output.

Per ulteriori informazioni sulle funzioni, consultare il [Capitolo 6, “Classi”](#)

Informazioni sulla continuazione delle animazioni mediante il metodo `continueTo()`

In “[Uso della classe Tween](#)” a pagina 537 viene illustrato come utilizzare la classe Tween nelle applicazioni. Tuttavia, se si desidera spostare la palla dopo il completamento dell'animazione iniziale, è possibile fare riferimento ad almeno due modalità. Una soluzione è costituita da una nuova animazione della palla mediante il gestore di eventi `onMotionFinished`. Tuttavia, la classe Tween offre una soluzione più semplice, vale a dire il metodo `continueTo()`. Il metodo `continueTo()` indica all'animazione interpolata di continuare dal valore corrente fino a un nuovo valore, come illustrato nel codice ActionScript seguente:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Object = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
    true);
ball_tween.onMotionFinished = function() {
    ball_tween.continueTo(0, 3);
};
```

Al termine dell'interpolazione iniziale, il clip filmato `ball_mc` torna alla sua posizione originale, 0 pixel. Nel frammento di codice seguente (abbreviato per ragioni di spazio), viene descritta la funzione di prototipo del metodo `continueTo()`.

```
function continueTo(finish:Number, duration:Number):Void {
    /* Omesso per ragioni di spazio. */
}
```

Solo due argomenti vengono passati al metodo `continueTo()` invece dei sette del metodo relativo alla funzione di costruzione della classe Tween, come illustrato nel frammento di codice seguente:

```
function Tween (obj, prop, func, begin, finish, duration, useSeconds) {
    /* Omesso per ragioni di spazio. */
}
```

I cinque parametri non richiesti dal metodo `continueTo()` sono `obj`, `prop`, `func`, `begin` e `useSeconds`. Nel metodo vengono usati gli argomenti definiti in precedenza nella chiamata alla classe Tween. Quando si chiama il metodo `continueTo()`, si presuppone che gli argomenti `obj`, `prop`, `func` (tipo di andamento) e `useSeconds` siano gli stessi della precedente chiamata alla classe Tween. Il metodo `continueTo()` usa il valore `finish` dalla chiamata alla classe Tween, invece di specificare un valore per l'argomento `begin`, come illustrato nel codice ActionScript seguente.

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Object = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
    true);
ball_tween.onMotionFinished = function() {
    ball_tween.continueTo(0, 3);
};
```

Questo codice sposta l'istanza `ball_mc` lungo l'asse `x` da 0 a 300 pixel in tre secondi. Al termine dell'animazione, viene attivato il gestore di eventi `onMotionFinished` e chiamato il metodo `continueTo()`. Il metodo `continueTo()` indica all'oggetto `target` (`ball_mc`) di animarsi partendo dalla posizione attuale per tre secondi lungo l'asse `x` a 0 pixel e di usare lo stesso metodo di andamento. I valori specificati nella chiamata al metodo della funzione di costruzione della classe Tween vengono usati per tutti i parametri non definiti nel metodo `continueTo()`. Se non si specifica una durata per il metodo `continueTo()`, viene applicata la durata indicata nella chiamata alla funzione di costruzione della classe Tween.

Creazione di immagini eseguite in modo continuo

È possibile fare in modo che un'animazione continui a spostarsi avanti e indietro lungo l'asse `x` senza fermarsi mai. La classe Tween svolge questo tipo di animazione mediante il metodo denominato `yoyo()`. Il metodo `yoyo()` attende l'esecuzione del gestore di eventi `onMotionFinished`, quindi inverte i parametri di `inizio` e `fine`. L'animazione viene avviata di nuovo, come dimostrato dalla procedura seguente.

Per creare un'animazione che continua all'infinito:

1. Creare un nuovo documento Flash denominato **yoyo.fla**.
2. Aprire il pannello Azioni e inserire il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
with (box_mc) {
    beginFill(0xFF0000, 60);
    moveTo(0, 0);
    lineTo(20, 0);
    lineTo(20, Stage.height);
    lineTo(0, Stage.height);
    lineTo(0, 0);
    endFill();
}
```

La prima sezione di codice precedente inizia con l'importazione della classe Tween e delle singole classi presenti nel pacchetto dell'andamento. La sezione di codice seguente crea una nuova istanza di clip filmato denominata **box_mc** e disegna un rettangolo largo 20 pixel e alto quanto lo stage.

3. Aggiungere il codice ActionScript seguente dopo il codice creato al punto precedente:

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
    Stage.width, 3, true);
box_tween.onMotionFinished = function() {
    box_tween.yoyo();
};
```

Questo codice consente di creare una nuova interpolazione per animare il clip filmato **box_mc** intorno allo stage lungo l'asse **x**- oltre 3 secondi.

4. Selezionare Controllo > Prova filmato per provare l'animazione.

Il riquadro viene animato da sinistra a destra e all'indietro. Se l'animazione non appare fluida, provare a incrementare la frequenza di fotogrammi del documento da 12 a 24 fps.

Man mano che si avvicina al bordo destro dello stage, il rettangolo prosegue l'animazione al di fuori del bordo. Anche se può sembrare ininfluente, il fatto che il rettangolo scompaia oltre il bordo dello stage e riappaia un secondo dopo per proseguire l'animazione in senso contrario potrebbe essere un effetto indesiderato.

In questi casi, eseguire l'animazione del rettangolo da 0 pixel alla larghezza dello stage meno la larghezza del clip filmato **box_mc**.

5. Per evitare che il rettangolo scompaia, rivedere le stringhe di codice corrispondenti dal passo 3 in modo da conformarsi al codice seguente:

```
var box Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,  
    (Stage.width - box_mc._width), 3, true);
```

6. Provare di nuovo l'animazione (Controllo > Prova filmato).

A questo punto, l'andamento del rettangolo si arresta prima che questo fuoriesca dal bordo dello stage.

Uso combinato delle classi Tween e TransitionManager

La combinazione delle classi Tween e TransitionManager consente di creare effetti interessanti. È possibile usare la classe TransitionManager per spostare un clip filmato lungo l'asse *x* mentre si esegue la regolazione della proprietà *_alpha* dello stesso clip mediante la classe Tween. Ogni classe può usare un metodo di andamento diverso, il che significa che esistono svariate possibilità di animazione degli oggetti nei file SWF. È possibile usare i metodi *continueTo()* e *yoyo()* nella classe Tween o il gestore di eventi *onMotionFinished* per creare un effetto davvero unico.

L'uso combinato delle classi TransitionManager e Tween consente di animare un clip filmato caricato in modo dinamico e dissolverlo sullo stage dopo che è stato completamente caricato dal server remoto, come illustrato nella procedura seguente.

Per usare le classi Tween e TransitionManager assieme:

1. Creare un nuovo documento Flash e salvare il file come **combination.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import mx.transitions.*;  
import mx.transitions.easing.*;  
  
var mcl_obj:Object = new Object();  
mcl_obj.onLoadInit = function(target_mc:MovieClip) {  
    new Tween(target_mc, "_alpha", Strong.easeIn, 0, 100, 2, true);  
    TransitionManager.start(target_mc, {type:Fly,  
        direction:Transition.IN, duration:3, easing:Elastic.easeInOut,  
        startPoint:6});  
};  
  
var my_mcl:MovieClipLoader = new MovieClipLoader();  
my_mcl.addListener(mcl_obj);  
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",  
    this.createEmptyMovieClip("img_mc", this.getNextHighestDepth()));
```

Questo codice è suddiviso in tre sezioni principali.

La prima importa le classi all'interno del pacchetto delle transizioni e il pacchetto `transitions.easing`. In questo esempio, il pacchetto delle transizioni viene importato interamente in modo da non dover immettere il nome completo delle classi Tween e TransitionManager o della transizione selezionata (in questo caso, *A entrata*). In questo modo, si riduce la lunghezza del codice che è necessario digitare diminuendo la possibilità di errori di ortografia.

La seconda sezione di ActionScript crea un oggetto listener per l'istanza di classe MovieClipLoader, che viene creata nella terza sezione del codice. Quando il clip filmato target viene caricato nell'istanza MovieClipLoader, l'evento `onLoadInit` attiva ed esegue il blocco di codice che chiama sia la classe Tween che la classe TransitionManager. Questo gestore di eventi si dissolve nel clip filmato target, in quanto viene modificata la proprietà `_alpha` della classe Tween, e applica un effetto *A entrata* al clip filmato lungo l'asse *x*.

La terza sezione di codice ActionScript crea un'istanza MovieClipLoader e applica l'oggetto listener creato in precedenza in modo che l'istanza loader del clip filmato intercetti l'evento `onLoadInit`. A questo punto, caricare l'immagine JPEG di destinazione in un clip filmato creato in modo dinamico chiamando il metodo `createEmptyMovieClip()`.

3. Salvare il documento e selezionare Controllo > Prova filmato per visualizzare l'animazione nell'ambiente di prova.

Dopo che l'immagine JPEG esterna è stata completamente scaricata dal server, l'immagine si dissolve gradualmente in entrata e viene animata da destra a sinistra sullo stage.

Per informazioni sull'utilizzo della classe Tween, consultare “[Uso della classe Tween](#)” [a pagina 537](#).

Per informazioni sui metodi e le proprietà della classe Tween, consultare il [Capitolo 51, “Classe Tween”](#) nella *Guida di riferimento dei componenti*. Per informazioni sui metodi e le proprietà della classe TransitionManager, consultare il [Capitolo 48, “Classe TransitionManager”](#) nella *Guida di riferimento dei componenti*.

È possibile trovare un file di origine di esempio che utilizza queste classi per aggiungere animazione con script. Individuare `tweenProgress.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a `unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar`.
- In Macintosh, accedere a `Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar`.

Uso degli effetti filtro

I filtri sono effetti visivi che è possibile applicare a oggetti a cui viene eseguito il rendering in fase di runtime, come le istanze di clip filmato. I filtri sono includono ombra esterna, sfocatura, bagliore, smussatura, bagliore con gradiente, smussatura con gradiente È anche possibile utilizzare una regola di colore che consenta di modificare la luminosità., il contrasto, la saturazione e la tonalità del clip filmato. È possibile applicare filtri mediante l'interfaccia utente Flash in Flash Professional 8 o mediante ActionScript in Flash Basic 8 o Flash Professional 8.

È possibile applicare tutti questi effetti di filtro a clip filmato, pulsanti o campi di testo mediante la scheda Filtri nella finestra di ispezione Proprietà o mediante ActionScript. Se si applicano filtri a un'istanza mediante ActionScript, è possibile utilizzare anche il filtro spostamento mappa (vedere “[Uso del filtro mappa di spostamento](#)” a pagina 577) o un filtro di convoluzione (vedere “[Uso del filtro di convoluzione](#)” a pagina 576). Questi filtri vengono applicati alle definizioni vettoriali, evitando così dati ridondanti derivanti dall'archiviazione di un'immagine bitmap in un file SWF. È anche possibile modificare un filtro esistente applicato a un campo di testo, clip filmato o pulsante mediante ActionScript.

Nella procedura seguente, viene illustrato come utilizzare un gestore di eventi `onEnterFrame` per animare un effetto di filtro bagliore in un clip filmato.

Per animare un effetto di filtro applicato a un'istanza di clip filmato:

1. Creare un nuovo documento Flash e salvarlo come `animFilter.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("box_mc", 10);
box_mc.lineStyle(20, 0x000000);
box_mc.beginFill(0x000000);
box_mc.moveTo(0, 0);
box_mc.lineTo(160, 0);
box_mc.lineTo(160, 120);
box_mc.lineTo(0, 120);
box_mc.lineTo(0, 0);
box_mc.endFill();
box_mc._x = 100;
box_mc._y = 100;

box_mc.filters = [new flash.filters.GlowFilter()];
var dir:Number = 1;
box_mc.blur = 10;
box_mc.onEnterFrame = function() {
    box_mc.blur += dir;
    if ((box_mc.blur >= 30) || (box_mc.blur <= 10)) {
        dir *= -1;
    }
}
```

```
var filter_array:Array = box_mc.filters;
filter_array[0].blurX = box_mc.blur;
filter_array[0].blurY = box_mc.blur;
box_mc.filters = filter_array;
};
```

Questo codice è associato a due diverse funzionalità. Nella prima sezione, viene creata e posizionata un'istanza di clip filmato e disegnato un rettangolo arrotondato nero sullo stage. Nel secondo blocco di codice, viene applicato un filtro di bagliore al rettangolo sullo stage e viene definito un gestore di eventi `onEnterFrame`, responsabile dell'animazione degli effetti di filtro. Il gestore di eventi `onEnterFrame` consente di animare il filtro di bagliore con una sfocatura tra 10 e 30 pixel. In seguito, l'animazione è pari o superiore a 3 o pari o inferiore a 10 e la direzione dell'animazione viene invertita.

3. Salvare le modifiche apportate al documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Per ulteriori informazioni sulle operazioni con i filtri in un'applicazione, consultare i seguenti argomenti:

- “[Operazioni con i pacchetti dei filtri](#)” a pagina 547
- “[Operazioni con i filtri, la memorizzazione nella cache e la classe MovieClip](#)” a pagina 549
- “[Informazioni su rilevamento per zone, rotazione, inclinazione e modifica in scala dei filtri](#)” a pagina 551
- “[Applicazione di filtri a istanze degli oggetti e istanze BitmapData](#)” a pagina 551
- “[Informazioni sulla gestione degli errori, sulle prestazioni e sui filtri](#)” a pagina 552

È presente un esempio di applicazione di filtri mediante ActionScript in un file di origine di esempio, Filters.fla., contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Filters*.

Operazioni con i pacchetti dei filtri

Pacchetti: directory che contengono uno o più file di classe e che risiedono nella directory del percorso di classe definita. Ad esempio, il pacchetto flash.filters è una directory sul disco rigido contenente diversi file di classi per ogni tipo di filtro (come BevelFilter, BlurFilter, DropShadowFilter e così via) in Flash 8. Quando questo tipo di file è organizzato in base a questa struttura, è necessario accedere alle classi con una modalità specifica. La classe viene *importata* o reperita mediante un *nome completo*.

NOTA

Per utilizzare l'istruzione import, è necessario specificare ActionScript 2.0 e Flash Player 6 o superiore nella scheda Flash della finestra di dialogo Impostazioni pubblicazione del file FLA.

L'istruzione `import` consente di accedere alle classi senza doverne specificare il nome completo. Ad esempio, per utilizzare un BlurFilter in uno script, è necessario reperire questo filtro mediante il nome completo (`flash.filters.BlurFilter`) o importarlo. Se viene importato, è possibile reperirlo mediante il nome di classe (`BlurFilter`). Nel codice ActionScript seguente, vengono illustrate le differenze tra l'utilizzo dell'istruzione `import` e l'utilizzo dei nomi di classe completi.

Se la classe `BlurFilter` non viene importata, è necessario utilizzare il nome di classe completo nel codice (nome del pacchetto seguito dal nome di classe) per utilizzare il filtro:

```
// senza importazione  
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10,  
3);
```

Lo stesso codice, scritto con un'istruzione `import`, consente di accedere al `BlurFilter` mediante il nome di classe anziché reperirlo utilizzando sempre il nome completo. In questo modo, la digitazione viene ridotta e diminuiscono le possibilità di commettere errori:

```
// mediante importazione  
import flash.filters.BlurFilter;  
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Per importare più classi in un pacchetto (tra le quali `BlurFilter`, `DropShadowFilter` e `GlowFilter`), è possibile fare ricorso a una delle due modalità di importazione delle classi. La prima modalità di importazione di classi multiple consiste nell'importare tutte le classi mediante un'istruzione `import` separata, come illustrato nel frammento di codice seguente:

```
import flash.filters.BlurFilter;  
import flash.filters.DropShadowFilter;  
import flash.filters.GlowFilter;
```

Se si utilizzano istruzioni import individuali per ogni classe in un pacchetto, la scrittura richiede molto tempo e aumenta la possibilità di commettere errori di digitazione. È possibile evitare di importare file di classi individuali mediante l'import *carattere jolly*, il quale consente di importare classi in un determinato livello di un pacchetto. Nel codice ActionScript seguente, viene presentato un esempio dell'utilizzo di un import carattere jolly:

```
import flash.filters.*; // importa ogni classe nel pacchetto flash.filters
```

L'istruzione `import` si applica solo allo script corrente (fotogramma o oggetto) in cui viene chiamata. Ad esempio, se nel fotogramma 1 di un documento Flash si importano tutte le classi presenti nel pacchetto `macr.util`. In quel fotogramma, è possibile reperire le classi nel pacchetto mediante i nomi di classi anziché il nome completo. Per utilizzare il nome di classe in un altro script di fotogrammi, fare riferimento alle classi nel pacchetto specificato mediante i nomi completi o aggiungere un'istruzione `import` nel fotogramma che consenta di importare classi nel pacchetto specificato.

Quando si utilizzano istruzioni `import`, bisogna tener presente che le classi sono importate soltanto per il livello specificato. Ad esempio, se si importano tutte le classi nel pacchetto `mx.transitions`, vengono importate soltanto le classi nella directory `/transitions/`, non quelle incluse nelle sottodirectory (come le classi nel pacchetto `mx.transitions.easing`).

SUGGERIMENTO

Se si importa una classe ma non la si utilizza nello script, la classe non viene esportata nel file SWF. Questo significa che è possibile importare pacchetti di grandi dimensioni senza doversi preoccupare delle dimensioni del file SWF; il codice byte associato a una classe viene incluso in un file SWF solo se tale classe viene effettivamente utilizzata.

È presente un esempio di applicazione di filtri mediante ActionScript in un file di origine di esempio, `Filters.fla`, contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Filters*.

Operazioni con i filtri, la memorizzazione nella cache e la classe MovieClip

Quando viene caricato il file SWF, se a un clip filmato è stato associato un filtro, esso viene contrassegnato in modo che sia memorizzato nella cache come una bitmap trasparente. Se al clip filmato è stato applicato almeno un filtro, Flash Player memorizza nella cache il clip filmato come una bitmap in fase di runtime, mantenendo la proprietà `cacheAsBitmap` su `true`. Questa bitmap memorizzata nella cache viene utilizzata come immagine di origine per gli effetti filtro. Ogni clip filmato dispone in genere di due bitmap: una bitmap è costituita dal clip filmato di origine non filtrata, mentre la seconda dall'immagine finale dopo il filtraggio. Se non si modifica l'aspetto del clip filmato in fase di runtime, non è necessario aggiornare l'immagine finale. In questo modo, si ottimizzano le prestazioni.

È possibile accedere ai filtri applicati a un'istanza chiamando la proprietà `MovieClip.filters`. La chiamata di questa proprietà restituisce un array contenente tutti gli oggetti filtro associati all'istanza di clip filmato. Ai filtri sono associate proprietà specifiche di quello stesso filtro, tra cui le seguenti:

```
trace(my_mc.filters[0].angle); // 45.0  
trace(my_mc.filters[0].distance); // 4
```

È possibile accedere ai filtri e modificarli come un normale oggetto array. L'impostazione e l'ottenimento dei filtri mediante la proprietà restituisce un *duplicato* dell'oggetto filtro, non un *riferimento*.

Per modificare un filtro esistente, è possibile utilizzare un codice simile a quello illustrato nella procedura seguente.

Per modificare le proprietà di un filtro applicato a un'istanza di clip filmato:

1. Creare un nuovo documento Flash e salvarlo come `modifyFilter.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("my_mc", 10);  
// Disegnare un quadrato  
with (my_mc) {  
    beginFill(0xFF0000, 100);  
    moveTo(0, 0);  
    lineTo(100, 0);  
    lineTo(100, 100);  
    lineTo(0, 100);  
    lineTo(0, 0);  
    endFill();  
}  
my_mc._x = 100;  
my_mc._y = 100;
```

```

// Utilizzare i valori DropShadowFilter predefiniti
my_mc.filters = [new flash.filters.DropShadowFilter()];
trace(my_mc.filters[0].distance); // 4
var filter_array:Array = my_mc.filters;
filter_array[0].distance = 10;
my_mc.filters = filter_array;
trace(my_mc.filters[0].distance); // 10

```

Nella prima sezione di questo codice, viene creato un quadrato rosso mediante l'API di disegno e posizionata la forma sullo stage. Nella seconda sezione di codice, viene applicato un filtro ombra esterna al quadrato. In seguito, viene creato un array temporaneo mediante codice per mantenere i filtri correnti in modo da applicare il quadrato rosso sullo stage. La proprietà `distance` del primo filtro è impostata su 10 pixel, mentre il filtro modificato viene applicato di nuovo all'istanza di clip filmato `my_mc`.

3. Selezionare Controllo > Prova filmato per provare il documento.

NOTA

Al momento, non è disponibile nessun supporto per eseguire la rotazione dei filtri basata sulla rotazione principale o su un altro tipo rotazione. Il filtro sfocatura consente di sfocare perfettamente sia in orizzontale che in verticale, indipendentemente dalla rotazione o dall'inclinazione di un qualsiasi elemento presente nella struttura ad albero dell'oggetto.

SUGGERIMENTO

Il contenuto filtrato è soggetto alle stesse limitazioni legate alla dimensione del contenuto con la proprietà `cacheAsBitmap` impostata su `true`. Se l'autore esegue un ingrandimento troppo distante sul file SWF, i filtri non sono più visibili qualora la rappresentazione della bitmap fosse più grande di 2800 pixel in entrambe le direzioni. Quando si pubblicano file SWF con filtri, è consigliabile disattivare le opzioni relative al menu di riduzione/ingrandimento.

È presente un esempio di applicazione di filtri mediante ActionScript in un file di origine di esempio, `Filters.fla`, contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Filters*.

Informazioni su rilevamento per zone, rotazione, inclinazione e modifica in scala dei filtri

Nella superficie relativa agli obiettivi del rilevamento per zone (operazione che consente di determinare se un'istanza si sovrappone o si interseca con un'altra) non sono presenti regioni filtrate (ad esempio, ombra esterna) al di fuori del rettangolo del riquadro di delimitazione dell'istanza di clip filmato. Siccome il rilevamento per zone si basa sui vettori, non è possibile eseguirlo sulla bitmap risultante. Ad esempio, se si applica un filtro smussatura all'istanza di un pulsante, non è possibile eseguire il rilevamento per zone sulla parte smussata dell'istanza.

La modifica in scala, la rotazione e l'inclinazione non sono supportate da filtri. Se l'istanza viene modificata in scala (`_xscale` e `_yscale` non sono 100%), l'effetto di filtro non viene modificato in scala con l'istanza. In altre parole, la forma originale dell'istanza viene ruotata, modificata in scala o inclinata; tuttavia, il filtro non viene ruotato, modificato in scala o inclinato con l'istanza.

È possibile animare un'istanza con un filtro per creare effetti realistici o nidificare istanze e utilizzare la classe `BitmapData` per animare i filtri in modo da ottenere l'effetto specificato.

Applicazione di filtri a istanze degli oggetti e istanze `BitmapData`

L'uso dei filtri dipende dall'istanza dell'oggetto al quale il filtro viene applicato. Seguire le istruzioni seguenti qualora si desiderasse applicare un filtro a un oggetto o un'istanza `BitmapData`:

- Per applicare i filtri a clip filmato, campi di testo e pulsanti in fase di runtime, utilizzare la proprietà `filters`. L'impostazione della proprietà `filters` di un oggetto non modifica l'oggetto e può essere annullata cancellando la proprietà `filters`.
- Per applicare filtri a istanze `BitmapData`, utilizzare il metodo `BitmapData.applyFilter()`. La chiamata di `applyFilter()` su un oggetto `BitmapData` modifica l'oggetto e non può essere annullata.

NOTA

(Soltanto Flash Professional 8) È anche possibile applicare effetti di filtro a immagini e video durante la fase di creazione mediante la scheda Filtri nella finestra di ispezione Proprietà.

Informazioni sulla gestione degli errori, sulle prestazioni e sui filtri

Se si applicano troppi filtri a un'applicazione, è necessario utilizzare una grande quantità di memoria, provocando così problemi alle prestazioni di Flash Player. Siccome un clip filmato con filtri associati dispone di due bitmap entrambe di 32 bit, l'utilizzo di molte bitmap potrebbe richiedere una grande quantità di memoria per l'applicazione. È possibile che venga visualizzato un errore legato all'esaurimento di memoria generato dal sistema operativo del computer. Errori simili non dovrebbero verificarsi con i computer moderni, anche se si utilizzano molti effetti di filtro in un'applicazione (ad esempio, qualora si disponesse di migliaia di bitmap sullo stage).

Tuttavia, se viene riscontrato un errore legato a esaurimento di memoria, si verificano gli eventi seguenti:

- L'array dei filtri viene ignorato.
- Il clip filmato viene disegnato con il normale renderer di vettori.
- I bitmap del clip filmato non vengono memorizzati nella cache.

Quando si verifica un errore legato a esaurimento di memoria, un clip filmato tenta di utilizzare un array di filtri o una cache delle bitmap. Un altro fattore che influenza sulle prestazioni dell'applicazione è il valore associato al parametro *quality* di ogni filtro applicato. Valori più elevati richiedono una maggiore quantità di CPU e memoria per l'effetto di rendering, mentre se si imposta il parametro *quality* su un valore più basso, sono richieste minori risorse del computer. Pertanto, evitare di utilizzare un numero eccessivo di filtri e, quando possibile, utilizzare un'impostazione *quality* più bassa.

ATTENZIONE

Se un oggetto di 100 x 100 pixel viene ingrandito una volta, la memoria viene utilizzata quattro volte, in quanto le dimensioni del contenuto, in questo caso, diventano 200 x 200 pixel. Se si ingrandisce altre due volte, la forma dell'oggetto passa a 800 x 800, utilizzando 64 volte in più la memoria rispetto all'oggetto originale di 100 x 100 pixel. Qualora fossero utilizzati filtri in un file SWF, è consigliabile disattivare le opzioni relative al menu di riduzione/ingrandimento nel menu di scelta rapida del file SWF.

In caso di parametri non validi, è possibile che si verifichino errori. Alcuni parametri del filtro hanno anche un intervallo valido particolare. Se viene impostato un valore al di fuori dell'intervallo valido, esso viene impostato su un valore valido compreso nell'intervallo. Ad esempio, *quality* dovrebbe essere compreso tra 1 e 3 in una comune operazione e può essere impostato soltanto su un valore compreso tra 0 e 15. Tutti i valori superiori a 15 vengono impostati su 15.

Inoltre, alcune funzioni di costruzione sono soggette a limitazioni sulla lunghezza degli array utilizzati come parametri di input. Se si realizza un filtro di convoluzione o un filtro matrice colore con un array non valido (dimensione non corretta), la funzione di costruzione non viene eseguita e il filtro non viene creato in modo corretto. Se l'oggetto dei filtri viene utilizzato come voce di un array dei filtri di un clip filmato, esso viene ignorato.

SUGGERIMENTO

Quando si utilizza un filtro di sfocatura, il ricorso a valori per blurX e blurY, proprietà alla potenza di 2 (come 2, 4, 8, 16 e 32), consente un'elaborazione più rapida e migliora le prestazioni del 20-30%.

Operazioni con i filtri mediante ActionScript

Il pacchetto flash.filters contiene classi per gli effetti di filtraggio delle immagini bitmap disponibili in Flash Player 8. I filtri consentono di applicare diversi effetti visivi mediante ActionScript, tra cui la sfocatura, la smussatura e l'ombra esterna al testo, al clip filmato e alle istanze del pulsante. È anche possibile utilizzare lo strumento di creazione di Flash per applicare effetti di filtro a oggetti quali testo, immagini e video. Flash dispone di nove effetti di filtro. Tuttavia, è possibile accedere soltanto a sette di essi mediante l'interfaccia utente di Flash Professional 8. I filtri ConvolutionFilter e DisplacementMapFilter sono disponibili soltanto mediante codice ActionScript.

NOTA

Tutti i filtri sono disponibili in ActionScript sia in Flash Basic 8 che in Flash Professional 8.

La procedura seguente consente di caricare un'immagine PNG semitrasparente e di applicare un effetto GlowFilter alla parte non trasparente dell'immagine.

Per applicare filtri a immagini semitrasparenti:

1. Creare un nuovo documento Flash e salvarlo come **transparentImg.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.GlowFilter;
System.security.allowDomain("http://www.helppexamples.com");
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
    target_mc._x = (Stage.width - target_mc._width) / 2;
    target_mc._y = (Stage.height - target_mc._height) / 2;
    var glow:GlowFilter = new GlowFilter();
    target_mc.filters = [glow];
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helppexamples.com/flash/images/logo.png",
    img_mc);
```

In questo codice, viene utilizzata un'istanza loader del clip filmato per caricare un'immagine PNG semitrasparente. Una volta caricata l'immagine, essa viene spostata al centro dello stage e viene applicato un filtro bagliore.

3. Selezionare Controllo > Prova filmato per provare il documento.

L'effetto bagliore viene applicato soltanto alla parte opaca (non trasparente) dell'immagine PNG.

Nella sezione seguente, viene descritto come utilizzare i filtri:

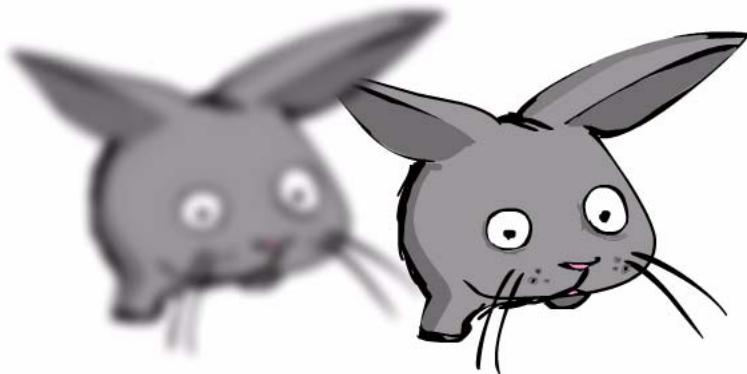
- “Uso del filtro sfocatura” a pagina 555
- “Uso del filtro ombra esterna” a pagina 557
- “Uso del filtro bagliore” a pagina 562
- “Creazione di bagliori con gradiente” a pagina 563
- “Uso del filtro smussatura” a pagina 565
- “Applicazione di un filtro smussatura con gradiente” a pagina 572
- “Uso del filtro matrice colore” a pagina 573
- “Uso del filtro di convoluzione” a pagina 576
- “Uso del filtro mappa di spostamento” a pagina 577

È presente un esempio di applicazione di filtri mediante ActionScript in un file di origine di esempio, Filters.fla., contenuto nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Filters*.

Uso del filtro sfocatura

La classe `BlurFilter` consente di applicare un effetto di sfocatura a numerosi oggetti in Flash. L'effetto sfocatura attenua i dettagli di un'immagine. È possibile realizzare sfocature molto diverse, da quelle che producono un effetto leggermente sfuocato fino alle sfocature gaussiane, che hanno un aspetto offuscato come quando si osserva un'immagine attraverso un vetro semiopaco. L'effetto sfocatura si basa su un filtro di sfocatura di tipo box-pass. Il parametro `quality` consente di definire il numero di volte in cui ripetere la sfocatura (il valore 3 si avvicina a un filtro sfocatura gaussiano).



NOTA

Il filtro sfocatura viene modificato in scala soltanto quando si ingrandisce nello stage.

Per ulteriori informazioni su questo filtro, consultare `BlurFilter` (`flash.filters.BlurFilter`) nella *Guida di riferimento di ActionScript 2.0*.

La procedura seguente consente di sfuocare un'immagine caricata in modo dinamico in base alla posizione corrente del puntatore del mouse sullo stage. Più il puntatore viene allontanato dal centro dello stage, più l'immagine viene sfuocata.

Per sfuocare un'immagine in base alla posizione del puntatore del mouse:

1. Creare un nuovo documento Flash e salvarlo come **dynamicblur.fla**.

2. Aggiungere il codice seguente al fotogramma 1 della linea temporale:

```
import flash.filters.BlurFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
    // Centrare il clip filmato target_mc sullo stage.
    target_mc._x = (Stage.width - target_mc._width) / 2;
    target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);
var blur:BlurFilter = new BlurFilter(10, 10, 2);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
    /* Moving the pointer to the center of the Stage sets the blurX and
    blurY properties to 0%. */
    blur.blurX = Math.abs(_xmouse - (Stage.width / 2)) / Stage.width * 2 *
    255;
    blur.blurY = Math.abs(_ymouse - (Stage.height / 2)) / Stage.height * 2 *
    255;
    img_mc.filters = [blur];
};
Mouse.addListener(mouseListener);
```

La prima sezione di questo codice carica e posiziona un'immagine caricata in modo dinamico sullo stage. Nella seconda sezione, viene definito un listener chiamato ogni qualvolta si sposta il mouse. La quantità di sfocatura orizzontale e verticale viene stabilita in base alla posizione corrente del puntatore del mouse sullo stage. Più il puntatore viene allontanato dal centro dello stage, più l'istanza viene sfuocata.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Spostare il puntatore del mouse lungo l'asse *x* per modificare la quantità di sfocatura orizzontale. L'istanza viene sfuocata maggiormente quando il puntatore viene allontanato dal centro orizzontale dello stage. Lo spostamento del puntatore lungo l'asse *y* provoca l'aumento o la diminuzione della sfocatura verticale, in base alla distanza dal centro verticale dello stage.

SUGGERIMENTO

Quando si applica un filtro sfocatura, l'utilizzo di valori per blurX e blurY, proprietà alla potenza di 2 (come 2, 4, 8, 16 e 32), consente un'elaborazione più rapida e migliora le prestazioni del 20-30%.

ATTENZIONE

L'impostazione di un valore di sfocatura più basso di 1,03125 comporta la disattivazione dell'effetto sfocatura.

Uso del filtro ombra esterna

La classe DropShadowFilter consente di aggiungere un'ombra esterna a una serie di oggetti in Flash. L'algoritmo dell'ombra si basa sullo stesso filtro box utilizzato dal filtro sfocatura (vedere “[Uso del filtro sfocatura](#)” a pagina 555). Sono disponibili diverse opzioni per lo stile dell'ombra esterna, tra cui l'ombra interna ed esterna e la modalità di foratura.

Per ulteriori informazioni su questo filtro, consultare DropShadowFilter (flash.filters.DropShadowFilter) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene disegnato un quadrato sullo stage mediante l'API di disegno. Quando si sposta il puntatore del mouse in orizzontale lungo lo stage, questo codice consente di modificare la distanza dal quadrato in cui viene visualizzata l'ombra esterna, mentre se si sposta il cursore in verticale, si modifica la sfocatura dell'ombra esterna.

Per utilizzare il filtro ombra esterna:

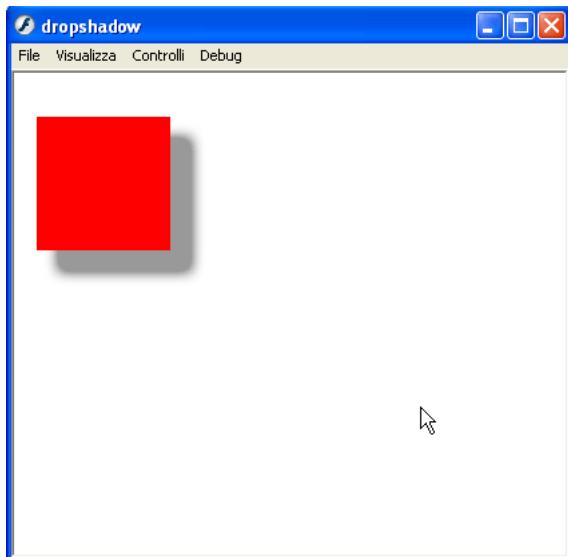
1. Creare un nuovo documento Flash e salvarlo come **dropshadow.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// import the filter classes
import flash.filters.DropShadowFilter;
// Creare un clip filmato chiamato shapeClip
this.createEmptyMovieClip("shapeClip", 1);
// Disegnare/creare una forma mediante l'API di disegno.
with (shapeClip) {
    beginFill(0xFF0000, 100);
    moveTo(0, 0);
    lineTo(100, 0);
    lineTo(100, 100);
    lineTo(0, 100);
    lineTo(0, 0);
    endFill();
}
// Posizionare la forma
shapeClip._x = 100;
shapeClip._y = 100;
// Fare clic sul quadrato e aumentare l'intensità dell'ombra
shapeClip.onPress = function():Void {
    dropShadow.strength++;
    shapeClip.filters = [dropShadow];
};
// Creare un filtro
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
    0.4, 10, 10, 2, 3);

var mouseListener:Object = new Object();
// Creare e applicare un listener che consenta di controllare il filtro
// quando il mouse viene spostato
mouseListener.onMouseMove = function():Void {
    dropShadow.distance = (_xmouse / Stage.width) * 50 - 20;
    dropShadow.blurX = (_ymouse / Stage.height) * 10;
    dropShadow.blurY = dropShadow.blurX;
    shapeClip.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

Nella prima sezione del codice, viene creato un nuovo clip filmato e disegnato un quadrato rosso mediante l'API di disegno. Nella seconda sezione, viene definito un listener del mouse, chiamato ogni qualvolta viene spostato il mouse. Il listener del mouse calcola la distanza dell'ombra esterna e il livello di sfocatura basato sulle posizioni *x* e *y* correnti del puntatore del mouse e applica di nuovo il filtro ombra esterna. Se si fa clic sul quadrato rosso, viene aumentata l'intensità dell'ombra esterna.

- 3.** Selezionare Controllo > Prova filmato per provare il documento Flash.
Spostare il puntatore del mouse lungo l'asse *x* per modificare il valore relativo alla distanza dell'ombra esterna e spostare il puntatore del mouse lungo l'asse *y* per modificare la quantità di sfocatura applicata all'istanza di clip filmato.



È anche possibile creare ombre esterne e applicarle a immagini caricate in modo dinamico. Nella procedura seguente, viene descritto come caricare un'immagine esterna e applicare un'ombra esterna che segua il puntatore del mouse. Più il puntatore viene allontanato dall'angolo superiore sinistro dell'immagine, più essa viene sfocata in orizzontale e in verticale.

Per creare un'ombra esterna che segua il puntatore del mouse:

1. Creare un nuovo documento Flash e salvarlo come **dropshadowmouse.fla**.
 2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:
- ```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
 0.8, 10, 10, 2, 2);
// Caricare e posizionare l'immagine sullo stage.
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip):Void {
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 img_mc);
```

```
// Quando il mouse viene spostato, calcolare nuovamente la posizione
// dell'ombra esterna.
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 var p1:Number = img_mc._y - _ymouse;
 var p2:Number = img_mc._x - _xmouse;
 var degrees:Number = Math.atan2(p1, p2) / (Math.PI / 180);
 dropShadow.distance = Math.sqrt(Math.pow(p1, 2) + Math.pow(p2, 2)) *
 0.5;
 dropShadow.blurX = dropShadow.distance;
 dropShadow.blurY = dropShadow.blurX;
 dropShadow.angle = degrees - 180;
 img_mc.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

Nella prima sezione di questo codice, viene definita un'istanza di ombra esterna, caricata un'immagine esterna e l'immagine viene riposizionata al centro dello stage. Nella seconda sezione di codice, viene definito un listener del mouse, chiamato ogni qualvolta si sposta il puntatore del mouse intorno allo stage. Ogni qualvolta si sposta il mouse, il gestore di eventi calcola nuovamente la distanza e l'angolazione tra il puntatore del mouse e l'angolo superiore sinistro dell'immagine. In base a questo calcolo, il filtro ombra esterna viene applicato nuovamente al clip filmato.

**3.** Selezionare Controllo > Prova filmato per provare il documento Flash.

Quando viene eseguito il file SWF, l'ombra esterna segue il puntatore del mouse. Più si avvicina il puntatore del mouse all'angolo superiore sinistro dell'immagine sullo stage, meno l'immagine viene sfuocata. In caso contrario, l'effetto di ombra esterna risulta più evidente.

È anche possibile applicare ombre esterne a immagini PNG semitrasparenti caricate in modo dinamico. Nella procedura seguente, il filtro ombra esterna viene applicato soltanto all'area piena dell'immagine PNG, non alla trasparenza.

**Per applicare un filtro ombra esterna a un'immagine semitrasparente:**

- 1.** Creare un nuovo documento Flash e salvarlo come **dropshadowTransparent.fla**.
- 2.** Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip):Void {
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
 var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45,
 0x000000, 0.5, 10, 10, 2, 3);
 target_mc.filters = [dropShadow];
};
mcListener.onLoadError = function(target_mc:MovieClip):Void {
 trace("unable to load image.");
};
this.createEmptyMovieClip("logo_mc", 10);
var my_mc:MovieClipLoader = new MovieClipLoader();
my_mc.addListener(mcListener);
my_mc.loadClip("http://www.helpexamples.com/flash/images/logo.png",
 logo_mc);
```

In questo codice di ActionScript, viene utilizzata la classe MovieClipLoader per caricare un'immagine e applicare un filtro ombra esterna una volta caricata l'immagine dal server remoto.

**3.** Selezionare Controllo > Prova filmato per provare il documento Flash.

Flash carica un'immagine PNG con uno sfondo trasparente. Il filtro ombra esterna viene applicato soltanto alla parte opaca (non trasparente) dell'immagine.

## Uso del filtro bagliore

La classe `GlowFilter` consente di aggiungere un effetto di bagliore a vari oggetti in Flash. L'algoritmo del bagliore si basa sullo stesso filtro box utilizzato dal filtro sfocatura (vedere [“Uso del filtro sfocatura” a pagina 555](#)). Sono disponibili diverse opzioni per lo stile di bagliore, tra cui il bagliore interno ed esterno e la modalità di foratura. Il filtro bagliore è molto simile al filtro ombra esterna, ma con le proprietà della distanza e dell'angolazione dell'ombra esterna impostate su 0.

Per ulteriori informazioni su questo filtro, consultare `GlowFilter` (`flash.filters.GlowFilter`) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene descritto come applicare un filtro bagliore a un clip filmato creato in modo dinamico sullo stage. Lo spostamento del puntatore del mouse intorno allo stage comporta la modifica della sfocatura del clip filmato e facendo clic sulla forma creata in modo dinamico, aumenta l'intensità del filtro.

### Per utilizzare il filtro bagliore:

1. Creare un nuovo documento Flash e salvarlo come `glowfilter.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.GlowFilter;

this.createEmptyMovieClip("shapeClip", 10);
with (shapeClip) {
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 100);
 lineTo(0, 100);
 lineTo(0, 0);
 endFill();
}
shapeClip._x = 100;
shapeClip._y = 100;
shapeClip.onPress = function():Void {
 glow.strength++;
 shapeClip.filters = [glow];
};
var glow:GlowFilter = new GlowFilter(0xCC0000, 0.5, 10, 10, 2, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 glow.blurX = (_xmouse / Stage.width) * 255;
 glow.blurY = (_ymouse / Stage.width) * 255;
 shapeClip.filters = [glow];
};
Mouse.addListener(mouseListener);
```

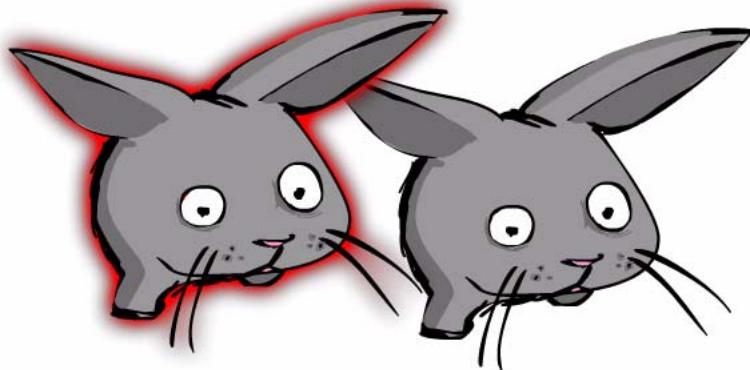
In questo codice, viene disegnato un quadrato sullo stage mediante l'API di disegno e applicato un filtro bagliore alla forma. Ogni qualvolta si sposta il puntatore del mouse lungo l'asse *x* o l'asse *y*, la sfocatura del filtro bagliore viene calcolata e applicata alla forma.

**3.** Selezionare Controllo > Prova filmato per provare il documento.

La quantità di sfocatura orizzontale e verticale viene calcolata in base alla posizione `_xmouse` e `_ymouse` corrente del puntatore del mouse. Se il puntatore del mouse viene spostato verso l'angolo superiore sinistro dello stage, diminuisce la quantità di sfocatura orizzontale e verticale. Al contrario, se il puntatore del mouse viene spostato verso l'angolo inferiore destro, aumenta la quantità di sfocatura orizzontale e verticale.

## Creazione di bagliori con gradiente

La classe `GradientGlowFilter` consente di creare un effetto di bagliore a gradiente per una serie di oggetti in Flash. Un bagliore a gradiente è un tipo di bagliore dall'aspetto realistico che è possibile applicare attorno al bordo interno o esterno di un oggetto o sopra di esso.



Per ulteriori informazioni su questo filtro, consultare `GradientBevelFilter` (`flash.filters.GradientBevelFilter`) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene disegnato un quadrato sullo stage mediante l'API di disegno e applicato un filtro bagliore con gradiente alla forma. Facendo clic sul quadrato sullo stage, si aumenta l'intensità del filtro, mentre se si sposta il puntatore del mouse in orizzontale o in verticale, si modifica la quantità di sfocatura lungo l'asse *x* o l'asse *y*.

**Per applicare un filtro bagliore con gradiente:**

1. Creare un nuovo documento Flash e salvarlo come **gradientglow.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.GradientGlowFilter;
// Creare una nuova istanza shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 10);
// Creare una forma mediante l'API di disegno.
with (shapeClip) {
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 100);
 lineTo(0, 100);
 lineTo(0, 0);
 endFill();
}

// Posizionare la forma
shapeClip._x = 100;
shapeClip._y = 100;
// Specificare un bagliore con gradiente
var gradientGlow:GradientGlowFilter = new GradientGlowFilter(0, 45,
 [0x000000, 0xFF0000], [0, 1], [0, 255], 10, 10, 2, 3, "outer");

// Definire un listener del mouse per due eventi
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
 gradientGlow.strength++;
 shapeClip.filters = [gradientGlow];
};
mouseListener.onMouseMove = function():Void {
 gradientGlow.blurX = (_xmouse / Stage.width) * 255;
 gradientGlow.blurY = (_ymouse / Stage.height) * 255;
 shapeClip.filters = [gradientGlow];
};
Mouse.addListener(mouseListener);
```

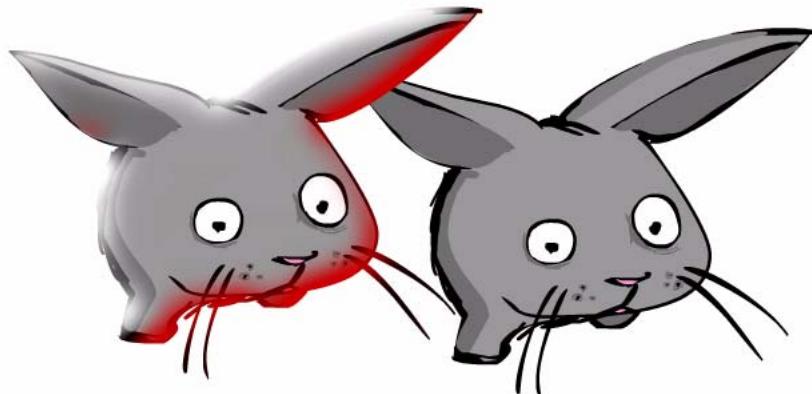
Il codice precedente viene suddiviso in tre sezioni. Nella prima sezione di questo codice, viene creato un quadrato mediante l'API di disegno e posizionata la forma sullo stage. Nella seconda sezione del codice, viene definita una nuova istanza del filtro bagliore con gradiente, nella quale viene creato un bagliore dal rosso al nero. Nella terza sezione del codice, viene definito un listener del mouse per due gestori di eventi del mouse. Il primo gestore di eventi è costituito da `onMouseDown`, il quale comporta l'aumento dell'intensità del bagliore con gradiente. Il secondo gestore di eventi è costituito da `onMouseMove`, il quale viene chiamato ogni qualvolta si sposta il puntatore del mouse nel file SWF. Più il puntatore del mouse viene allontanato dall'angolo superiore sinistro del documento Flash, più l'effetto bagliore risulta intenso.

3. Selezionare Controllo > Prova filmato per provare il documento.

Quando si sposta il puntatore del mouse intorno allo stage, la sfocatura del filtro bagliore con gradiente aumenta e l'intensità dimuisce. Fare clic col pulsante sinistro del mouse per aumentare l'intensità del bagliore.

## Uso del filtro smussatura

La classe `BevelFilter` consente di aggiungere un effetto smussatura a una serie di oggetti in Flash. L'effetto smussatura dona agli oggetti un aspetto tridimensionale. È possibile personalizzare l'aspetto dello smussatura mediante colori di evidenziazione e ombra diversi e modificando la quantità della sfocatura, l'angolo, la posizione della smussatura e l'effetto foratura.



Per ulteriori informazioni su questo filtro, consultare `BevelFilter` (`flash.filters.BevelFilter`) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene creato un quadrato mediante l'API di disegno e aggiunta una smussatura alla forma.

**Per utilizzare il filtro smussatura:**

1. Creare un nuovo documento Flash e salvarlo come **bevel.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.BevelFilter;
// Definire un filtro smussatura
var bevel:BevelFilter = new BevelFilter(4, 45, 0xFFFFF, 1, 0xCC0000, 1,
 10, 10, 2, 3);
// Creare una nuova istanza shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
// Creare una forma mediante l'API di disegno.
with (shapeClip) {
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 100);
 lineTo(0, 100);
 lineTo(0, 0);
 endFill();
}
// Posizionare la forma sullo stage
shapeClip._x = 100;
shapeClip._y = 100;
// Fare clic col mouse per aumentare l'intensità
shapeClip.onPress = function():Void {
 bevel.strength += 2;
 shapeClip.filters = [bevel];
};

// Definire un listener per modificare il filtro quando si sposta il
// mouse
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 bevel.distance = (_xmouse / Stage.width) * 10;
 bevel.blurX = (_ymouse / Stage.height) * 10;
 bevel.blurY = bevel.blurX;
 shapeClip.filters = [bevel];
};
Mouse.addListener(mouseListener);
```

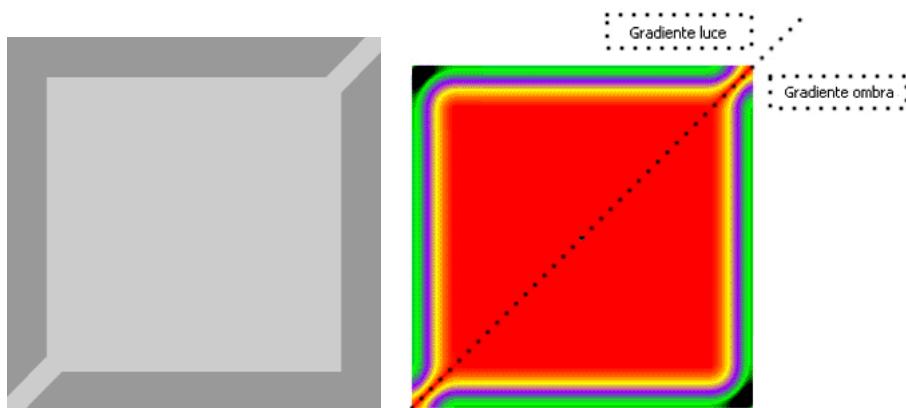
Nella prima sezione del codice, viene definita un'istanza BevelFilter e creato un quadrato sullo stage mediante l'API di disegno. Quando si fa clic sul quadrato dello stage, il valore di intensità corrente della smussatura viene aumentato e la smussatura acquista un aspetto più allungato e nitido. Nella seconda sezione del codice, viene definito un listener del mouse, il quale consente di modificare la distanza della smussatura e la sfocatura in base alla posizione corrente del puntatore del mouse.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Quando il puntatore del mouse viene spostato lungo l'asse *x*, la distanza offset della smussatura aumenta o diminuisce. Quando il puntatore del mouse viene spostato lungo l'asse *y*, le coordinate correnti del puntatore del mouse consentono di modificare la quantità della sfocatura orizzontale e verticale.

## Informazioni sul filtro smussatura con gradiente

Il filtro smussatura con gradiente viene applicato a oggetti quali un rettangolo, con i colori del gradiente distribuiti nelle tre parti del rettangolo: due bordi di smussatura (*evidenziazione* e *ombra*) e un'area denominata *riempimento di base*. Nei diagrammi seguenti, viene mostrato il rettangolo con il tipo di smussatura impostato all'interno. Nel rettangolo sulla sinistra, le aree grigio scuro costituiscono i bordi smussati, mentre le aree grigio chiaro rappresentano il riempimento di base. Nel rettangolo sulla destra, viene applicata una smussatura con gradiente multicolore costituita da quattro colori su ogni bordo.



Le diverse proprietà del filtro smussatura con gradiente consentono di controllare la modalità di applicazione del filtro. I colori della smussatura con gradiente vengono impostati nell'array dei colori. La distribuzione effettiva di colori in ogni parte del triangolo viene determinata dall'array ratios. La proprietà distance consente di determinare la distanza di offset o la quantità di pixel posta tra la posizione specificata e l'area dell'oggetto in cui viene applicato il bordo di smussatura. Le proprietà blurX e blurY consentono di controllare la precisione dei colori nella smussatura. La smussatura risulta più ampia e sfumata quando i valori sono più alti; in caso contrario, la smussatura risulta più sottile e nitida. La proprietà angle costituisce la sorgente di luce ipotetica sull'oggetto, ossia l'effetto evidenziazione e ombra sui bordi dell'oggetto. La proprietà strength consente di controllare l'applicazione dei colori: un valore più basso provoca la modifica dei colori, come nell'esempio; in caso contrario, i numeri esterni nell'array risultano più intensi, in quanto i colori centrali dell'array si notano di meno. Infine, le proprietà knockout e type consentono di determinare in che modo e in quale posizione viene applicato il filtro di smussatura all'interno dell'oggetto nel suo complesso: se il filtro nasconde l'oggetto e dove viene posizionato.

Uno degli aspetti più complessi nell'ambito dell'applicazione del filtro smussatura con gradiente è legato alla distribuzione del colore. Per distribuire correttamente i colori in una smussatura con gradiente, è necessario scegliere i colori che si desidera applicare in essa. Siccome i concetti di evidenziazione e ombra del colore in una smussatura sono già stati acquisiti, è possibile applicare gli stessi concetti al filtro smussatura con gradiente: si dispone di un gradiente di evidenziazione e un gradiente di ombra. Il primo viene visualizzato nell'angolo superiore sinistro, mentre il secondo nell'angolo inferiore destro. Nel gradiente relativo all'evidenziazione sono presenti quattro colori; lo stesso vale per il gradiente relativo all'ombra. Tuttavia, è necessario aggiungere un altro colore (il colore di riempimento di base), visualizzato nella parte in cui si incontrano i bordi di evidenziazione e d'ombra. Gli array dei colori dispongono di nove colori (è possibile visualizzarli nel diagramma precedente).

Il numero dei colori presente nell'array dei colori consente di determinare il numero di elementi negli array alphas e ratios. Il primo elemento nell'array dei colori corrisponde al primo elemento negli array alphas e ratios e così via. Siccome i colori sono nove, si dispone anche di nove valori nell'array alphas e di altri nove nell'array ratios. I valori alpha consentono di impostare il valore di trasparenza alpha dei colori.

I valori ratio nell'array ratios possono essere compresi tra 0 e 255 pixel. Il valore centrale è 128. 128 costituisce il valore del riempimento di base. Per la maggior parte degli utilizzi, per ottenere l'effetto smussatura desiderato, è consigliabile assegnare i valori ratio seguenti, come nell'esempio di nove colori:

- I primi quattro colori sono compresi tra 0 e 127, in base a un valore crescente, in modo che ogni valore sia pari o superiore al precedente. Si tratta del primo bordo di smussatura, ossia l'evidenziazione.
- Il quinto colore (il colore centrale) rappresenta la base di riempimento, impostata su 128. Il valore di pixel di 128 consente di impostare il riempimento di base, il quale può essere visualizzato al di fuori della forma (e intorno ai bordi della smussatura) se type è impostato all'esterno, oppure all'interno della forma, coprendo effettivamente il riempimento stesso dell'oggetto, se type è impostato all'interno.
- Gli ultimi quattro colori sono compresi tra 129 e 255, in base a un valore crescente, in modo che ogni valore sia pari o superiore al precedente. Si tratta del secondo bordo di smussatura, ad esempio l'ombra.

Se si desidera ottenere un gradiente composto da strisce di vario colore da fondere l'una nell'altra, ciascun valore ratio consente di impostare il numero di pixel del colore associato, ossia l'ampiezza della striscia del colore nel gradiente. Se si desidera distribuire i colori di ciascun bordo allo stesso modo:

- Utilizzare un numero dispari di colori, laddove il colore centrale costituisce il riempimento di base.
- Distribuire i valori compresi tra 0 e 127 e tra 129 e 255 in modo identico tra i colori.
- Regolare il valore in modo da modificare l'ampiezza di ciascuna striscia di colore nel gradiente.



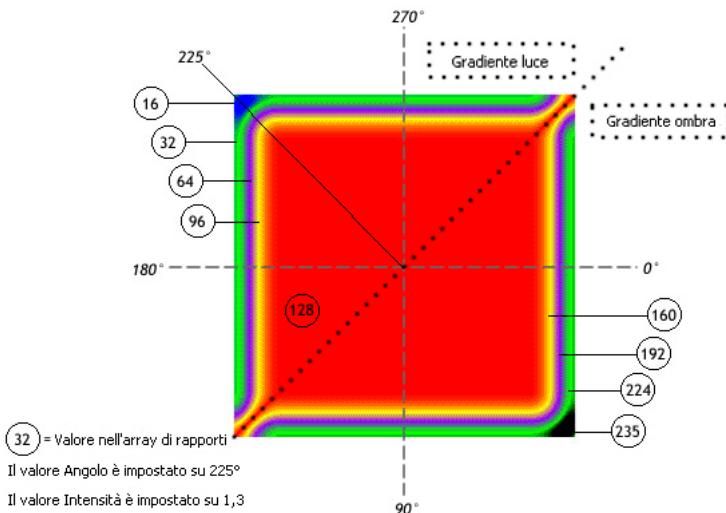
Il valore angle consente di determinare quale bordo rappresenta l'evidenziazione e quale l'ombra.

Il valore angle consente di determinare l'angolo dell'oggetto in cui vengono applicati i colori con gradiente, ossia la posizione in cui vengono visualizzati l'evidenziazione e l'ombra. I colori vengono applicati in base allo stesso ordine dell'array.

Nel codice seguente, viene utilizzato un quadrato rosa (disegnato mediante l'API di disegno) e applicato un filtro con gradiente multicolore. I colori, nell'ordine in cui sono presenti nell'array, sono: blu, verde, viola e giallo (evidenziazione); rosso (base di riempimento); giallo, viola, verde, nero (ombra). Per determinare i valori ratios, si assegnano quattro valori dei colori di evidenziazione compresi tra 0 e 127, rendendoli uguali in linea di massima e colori d'ombra compresi tra 129 e 255. I colori sul bordo esterno, blu (16) e nero (235).

```
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
 0xFFFF00, 0x9900FF, 0x00FF00, 0x000000];
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
 colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);
```

Nella figura seguente, viene mostrato il filtro smussatura con gradiente creato mediante il codice presentato sopra, una smussatura multicolore formata da nove colori applicata al clip filmato di un rettangolo rosso.



La linea tratteggiata mostra gli angoli specificati: Nella figura, viene mostrato l'angolo di 225° realizzato sul filtro e anche tutti i valori ratio di ciascun colore. L'impostazione dell'angolo a 225° indica che il primo colore nell'array inizia a 225°, nell'angolo superiore sinistro (l'evidenziazione). La linea punteggiata mostra la posizione in cui viene applicata la smussatura con gradiente e la posizione in cui viene applicato il gradiente di ombra.

Il colore del clip filmato originale è rosa, ma se si imposta il valore su rosso (128), il valore di 128 pixel costituisce il riempimento di base e copre il riempimento del clip filmato originale. Tuttavia, quando si imposta la proprietà filters, l'oggetto originale non viene alterato; se si elimina la proprietà filters, è possibile ripristinare il riempimento del clip filmato originale. Le proprietà di tutti i filtri influiscono sulle altre. In questo modo, se si regola una proprietà per modificare l'effetto che si sta applicando, è necessario modificare anche un'altra proprietà.

Per creare la figura precedente, utilizzare il seguente codice ActionScript:

```
import flash.filters.GradientBevelFilter;

// Disegna una forma quadrata piena
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF99CC);
square_mc.moveTo(40, 40);
square_mc.lineTo(200, 40);
square_mc.lineTo(200, 200);
square_mc.lineTo(40, 200);
square_mc.lineTo(40, 40);
square_mc.endFill();

/* GradientBevelFilter(distance:Number, angle:Number, colors:Array,
 alphas:Array, ratios:Array, blurX:Number, blurY:Number, strength:Number,
 quality:Number, type:String, knockout:Boolean) */

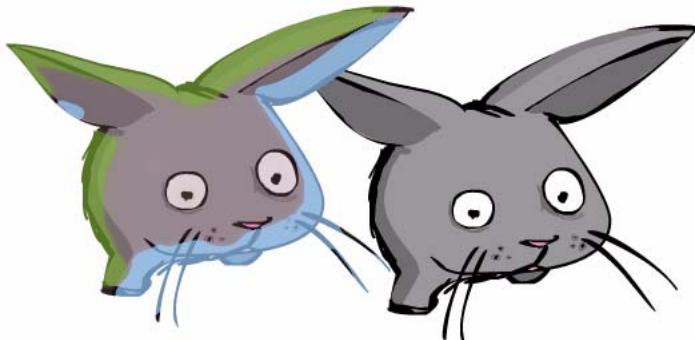
// Creare array colori, alphas e ratios
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
 0xFFFF00, 0x9900FF, 0x00FF00, 0x000000];//blue, green, purple, yellow,
 red, yellow, purple, green, black
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];

// Creare l'oggetto filter
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
 colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);

// Applica il filtro al clip filmato quadrato
square_mc.filters = [gradientBevel];
```

## Applicazione di un filtro smussatura con gradiente

La classe GradientBevelFilter consente di applicare un effetto di smussatura con gradiente a vari oggetti in Flash. Una smussatura con gradiente è un bordo smussato, ottimizzato con un gradiente di colore e applicato all'interno, all'esterno o sopra un oggetto. I bordi smussati forniscono un aspetto tridimensionale agli oggetti ed effetti di colore più vivaci come mostrato nella figura seguente:



Per ulteriori informazioni su questo filtro, consultare [GradientBevelFilter](#) (`flash.filters.GradientBevelFilter`) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene disegnato un quadrato sullo stage mediante l'API di disegno e applicato un filtro smussatura con gradiente alla forma.

### Per utilizzare il filtro smussatura con gradiente:

1. Creare un nuovo documento Flash e salvarlo come **gradientbevel.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.GradientBevelFilter;
var shapeClip:MovieClip = this.createEmptyMovieClip("shape_mc", 1);
with (shapeClip) {
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(200, 0);
 lineTo(200, 200);
 lineTo(0, 200);
 lineTo(0, 0);
 endFill();
}
shapeClip._x = (Stage.width - shapeClip._width) / 2;
shapeClip._y = (Stage.height - shapeClip._height) / 2;
var colors:Array = new Array(0xFFFFF, 0xCCCCC, 0x000000);
var alphas:Array = new Array(1, 0, 1);
var ratios:Array = new Array(0, 128, 255);
```

```

var gradientBevel:GradientBevelFilter = new GradientBevelFilter(10, 45,
 colors, alphas, ratios, 4, 4, 5, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
 gradientBevel.strength++;
 shapeClip.filters = [gradientBevel];
};
mouseListener.onMouseMove = function() {
 gradientBevel.blurX = (_xmouse / Stage.width) * 255;
 gradientBevel.blurY = (_ymouse / Stage.height) * 255;
 shapeClip.filters = [gradientBevel];
};
Mouse.addListener(mouseListener);

```

In questo codice, viene disegnato un quadrato sullo stage mediante l'API di disegno, il quale viene posizionato al centro dello stage. Quando si sposta il puntatore del mouse intorno allo stage, la quantità di sfocatura lungo l'asse *x* e l'asse *y* aumenta o diminuisce. Quando si sposta il puntatore verso la parte sinistra dello stage, diminuisce la quantità di sfocatura orizzontale. Quando si sposta il puntatore verso la parte destra dello stage, la sfocatura aumenta. In modo analogo, più il puntatore viene spostato verso l'alto, meno sfocatura è presente sull'asse *y*.

3. Selezionare Controllo > Prova filmato per provare il documento e visualizzare i risultati.

## Uso del filtro matrice colore

La classe ColorMatrixFilter consente di applicare una trasformazione di matrice 4 x 5 ai valori di colore ARGB e alfa di ogni pixel dell'immagine di input per produrre un risultato con una nuova serie di valori di colore ARGB e alfa. Questo filtro consente la rotazione della tonalità (colore diverso), le modifiche della saturazione (intensità di una tonalità specifica), la luminanza ad alfa (luminosità o intensità di un colore) e altri effetti vari. Inoltre, è possibile animare questi filtri per creare effetti nell'applicazione.

**NOTA**

È possibile applicare il filtro matrice colore alle immagini bitmap e alle istanze di clip filmato.

Per ulteriori informazioni su questo filtro, consultare ColorMatrixFilter (flash.filters.ColorMatrixFilter) nella *Guida di riferimento di ActionScript 2.0*.

È possibile utilizzare questo filtro per modificare la luminosità di un'istanza, come illustrato nell'esempio seguente.

### **Per aumentare la luminosità di un clip filmato:**

1. Creare un nuovo documento Flash e salvarlo come **brightness.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
 var myElements_array:Array = [1, 0, 0, 0, 100,
 0, 1, 0, 0, 100,
 0, 0, 1, 0, 100,
 0, 0, 0, 1, 0];
 var myColorMatrix_filter:ColorMatrixFilter = new
 ColorMatrixFilter(myElements_array);
 target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
 img_mc);
```

Questo codice consente di caricare un'immagine JPEG in modo dinamico mediante un'istanza MovieClipLoader. Una volta caricata l'immagine e posizionata sullo stage, la luminosità dell'istanza è impostata su 100% mediante un filtro matrice colore.

3. Selezionare Controllo > Prova filmato per provare il documento.

È anche possibile creare un effetto di luminosità animato associando la classe Tween alla classe ColorMatrixFilter, come illustrato nella procedura seguente.

### **Per animare il livello di luminosità di un'istanza mediante la classe Tween:**

1. Creare un nuovo documento Flash e salvarlo come **brightnesstween.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.ColorMatrixFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
 // Centra l'istanza di clip filmato sullo stage
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
 target_mc.watch("brightness", brightnessWatcher, target_mc);
 // Anima il clip filmato target_mc tra -100 e +100 di luminosità
 var t:Object = new Tween(target_mc, "brightness", Elastic.easeOut,
 100, -100, 3, true);
 t.onMotionFinished = function() {
```

```

 this.yoyo();
 };
};

this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 img_mc);

function brightnessWatcher(prop:String, oldVal:Number, newVal:Number,
 target_mc:MovieClip):Number {
 var brightness_array:Array = [1, 0, 0, 0, newVal,
 0, 1, 0, 0, newVal,
 0, 0, 1, 0, newVal,
 0, 0, 0, 1, 0];
 target_mc.filters = [new ColorMatrixFilter(brightness_array)];
 return newVal;
}

```

Nella prima sezione di codice, viene caricata un'immagine JPEG nello stage mediante la classe MovieClipLoader. Una volta caricata l'immagine, è necessario riposizionare l'immagine al centro dello stage. Quindi, è necessario utilizzare la classe Tween per animare il livello di luminosità dell'immagine. Per animare la luminosità, è necessario utilizzare il metodo `Object.watch()`, il quale consente di registrare un gestore di eventi avviato quando viene modificata una proprietà specifica dell'oggetto ActionScript. Ogni qualvolta un codice di ActionScript tenta di impostare la proprietà di luminosità personalizzata dell'istanza `target_mc`, viene chiamata la funzione `brightnessWatcher`. La funzione personalizzata `brightnessWatcher` consente di creare un nuovo array che imposta la luminosità dell'immagine di destinazione a una quantità specifica mediante un filtro matrice colore.

### **3. Selezionare Controllo > Prova filmato per provare il documento.**

Una volta caricata e posizionata l'immagine sullo stage, la luminosità dell'immagine viene animata tra -100 e 100. Una volta completata l'interpolazione della luminosità, l'animazione viene invertita con il metodo `Tween.yoyo()` che consente all'interpolazione di animarsi costantemente.

## Uso del filtro di convoluzione

La classe ConvolutionFilter applica un effetto di filtro di convoluzione matrice. Una convoluzione combina i pixel dell'immagine di origine specificata con i pixel adiacenti per produrre un'immagine. Grazie alle convoluzioni, è possibile effettuare una vasta gamma di operazioni sulle immagini, tra cui le sfocature, il rilevamento dei bordi, le modifiche della nitidezza, i rilievi e le smussature.

**NOTA**

Questo filtro può essere applicato alle istanze delle immagini bitmap e dei clip filmato.

La convoluzione matrice si basa su una matrice  $n$ -per- $m$ , che descrive il modo in cui un determinato valore di pixel nell'immagine di input viene combinato con i valori di pixel adiacenti per produrre un valore di pixel risultante. Ogni pixel risultante viene determinato dall'applicazione della matrice al pixel sorgente corrispondente e ai pixel a esso adiacenti.

Il filtro è disponibile soltanto mediante ActionScript. Per ulteriori informazioni su questo filtro, consultare ConvolutionFilter (flash.filters.ConvolutionFilter) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, il filtro di convoluzione viene applicato a un'immagine JPEG caricata in modo dinamico.

### Per utilizzare un filtro di convoluzione in modo da modificare il colore di un'immagine:

1. Creare un nuovo documento Flash e salvarlo come **convolution.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

this.createEmptyMovieClip("shape_mc", 1);
shape_mc.createEmptyMovieClip("holder_mc", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
 image1.jpg", shape_mc.holder_mc);
var matrixArr:Array = [1, 4, 6, 4, 1, 4, 16, 24, 16, 4, 16, 6, 24, 36,
 24, 6, 4, 16, 24, 16, 4, 1, 4, 6, 4, 1];
var convolution:ConvolutionFilter = new ConvolutionFilter(5, 5,
 matrixArr);
shape_mc.filters = [convolution];

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 convolution.divisor = (_xmouse / Stage.width) * 271;
 convolution.bias = (_ymouse / Stage.height) * 100;
```

```
 shape_mc.filters = [convolution];
};

Mouse.addListener(mouseListener);
```

Il codice precedente viene suddiviso in tre sezioni. Nella prima sezione, vengono importate due classi: ConvolutionFilter e BitmapData. Nella seconda sezione, viene creato un clip filmato nidificato e utilizzato un oggetto loader del clip filmato per caricare un'immagine nel clip filmato nidificato. Viene creato un oggetto del filtro di convoluzione, il quale viene applicato al clip filmato shape\_mc. Nella sezione finale del codice, viene definito un oggetto listener del mouse che consente di modificare le proprietà divisor o bias del filtro di convoluzione in base alla posizione corrente del puntatore del mouse e di riapplicare il filtro di convoluzione al clip filmato shape\_mc.

**3. Selezionare Controllo > Prova filmato per provare il documento Flash.**

Se si sposta il mouse lungo l'asse x dello stage, viene modificato il divisore (divisor) del filtro, mentre se si sposta il mouse lungo l'asse y dello stage, viene modificata la variazione (bias) del filtro.

## Uso del filtro mappa di spostamento

La classe DisplacementMapFilter utilizza i valori in pixel dell'oggetto BitmapData specificato (o *immagine della mappa di spostamento*) per eseguire uno spostamento di un'istanza che si trova sullo stage, ad esempio un'istanza di clip filmato o un'istanza di dati bitmap. Questo filtro può essere utilizzato per applicare un effetto deformato o screziato a un'istanza specificata.

Il filtro è disponibile soltanto mediante ActionScript. Per ulteriori informazioni su questo filtro, consultare DisplacementMapFilter (flash.filters.DisplacementMapFilter) nella *Guida di riferimento di ActionScript 2.0*.

Nella procedura seguente, viene caricata un'immagine JPEG e applicato un filtro mappa di spostamento a essa, in modo da farla risultare distorta. Ogni qualvolta si sposta il mouse, la mappa di spostamento viene rigenerata.

**Per distorcere un'immagine mediante il filtro mappa di spostamento:**

1. Creare un nuovo documento Flash e salvarlo come **displacement.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.display.BitmapData;

var perlinBmp:BitmapData;
var displacementMap:DisplacementMapFilter;
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip):Void {
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
 perlinBmp = new BitmapData(target_mc._width, target_mc._height);
 perlinBmp.perlinNoise(target_mc._width, target_mc._height, 10,
 Math.round(Math.random() * 100000), false, true, 1, false);
 displacementMap = new DisplacementMapFilter(perlinBmp, new Point(0,
 0), 1, 1, 100, 100, "color");
 shapeClip.filters = [displacementMap];
};

var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.addListener(mcListener);
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
 image1.jpg", shapeClip.holderClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 perlinBmp.perlinNoise(shapeClip._width, shapeClip._height, 10,
 Math.round(Math.random() * 100000), false, true, 1, false);
 shapeClip.filters = [displacementMap];
};
Mouse.addListener(mouseListener);
```

Questo codice consente di caricare un'immagine JPEG e di posizionarla sullo stage. Una volta caricata l'immagine, il codice crea un'istanza BitmapData e utilizza il metodo `perlinNoise()` per riempirla con pixel posizionati in maniera casuale. L'istanza BitmapData viene passata al filtro mappa di spostamento, il quale viene applicato all'immagine per farla risultare distorta.

**3.** Selezionare Controllo > Prova filmato per provare il documento.



Spostare il puntatore del mouse intorno allo stage per ricreare una mappa di spostamento mediante il metodo `perlinNoise()` che consente di modificare l'aspetto dell'immagine JPEG.

## Gestione degli effetti filtro mediante il codice

Flash Basic 8 e Flash Professional 8 consentono di aggiungere in modo dinamico vari filtri ai clip filmato, campi di testo e pulsanti sullo stage anziché filtri nell'ambiente di creazione di Flash Professional 8 (mediante la scheda Filtri nella finestra di ispezione Proprietà). Quando si inseriscono e modificano filtri nel corso della riproduzione, è possibile aggiungere ombre realistiche, sfocature e bagliori che rispondano al movimento del mouse o agli eventi dell'utente.

Per alcuni esempi relativi alla modifica dei filtri con codice, consultare gli argomenti seguenti:

- “Regolazione delle proprietà dei filtri” a pagina 580
- “Animazione di un filtro mediante ActionScript” a pagina 582
- “Uso del metodo `clone()`” a pagina 583

## Regolazione delle proprietà dei filtri

È possibile accedere all'array dei filtri applicati agli oggetti con chiamate ActionScript standard mediante la proprietà `MovieClip.filters`. Questo processo restituisce un array contenente tutti gli oggetti filtro correntemente associati a `MovieClip`. Ciascun filtro dispone di un insieme di proprietà specifiche di quel filtro. È possibile accedere ai filtri e modificarli come se si trattasse di un oggetto array. Tuttavia, l'ottenimento e l'impostazione di filtri mediante la proprietà `filters` restituisce un duplicato dell'oggetto filtri anziché un riferimento.

L'impostazione della proprietà `filters` consente di duplicare l'array dei filtri passati non archiviandoli come riferimento. Quando si ottiene la proprietà `filters`, viene restituita una nuova copia dell'array. In questo approccio, è presente un aspetto negativo: il codice seguente non funziona.

```
// Non funziona
my_mc.filters[0].blurX = 20;
```

Siccome il frammento di codice precedente restituisce una copia dell'array dei filtri, il codice modifica la copia anziché l'array originale. Per modificare la proprietà `blurX`, è necessario utilizzare il codice ActionScript seguente:

```
// Funziona
var filterArray:Array = my_mc.filters;
filterArray[0].blurX = 20;
my_mc.filters = filterArray;
```

La procedura seguente consente di sfuocare un'immagine in base alla posizione corrente del puntatore del mouse sullo stage. Ogni qualvolta il puntatore del mouse viene spostato in orizzontale o in verticale, le proprietà `blurX` e `blurY` del filtro sfocatura vengono modificate di conseguenza.

**Per regolare le proprietà del filtro del clip filmato:**

1. Creare un nuovo documento Flash e salvarlo come **adjustfilter.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.BlurFilter;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);
holder_mc.img_mc.loadMovie("http://www.helpexamples.com/flash/images/
 image2.jpg");
holder_mc.filters = [new BlurFilter(10, 10, 2)];
holder_mc._x = 75;
holder_mc._y = 75;

holder_mc.onMouseMove = function() {
 var tempFilter:BlurFilter = holder_mc.filters[0];
 tempFilter.blurX = Math.floor((_xmouse / Stage.width) * 255);
 tempFilter.blurY = Math.floor((_ymouse / Stage.height) * 255);
 holder_mc.filters = [tempFilter];
};
```

Il codice precedente viene suddiviso in tre sezioni. La prima sezione, consente di importare la classe `flash.filters.BlurFilter`. In questo modo, non è necessario utilizzare il nome di classe completo quando si fa riferimento alla classe `BlurFilter`. La seconda sezione consente di creare due clip filmato e di caricare un'immagine all'interno di uno dei clip nidificati. La terza sezione del codice risponde al movimento del mouse sullo stage e, di conseguenza, consente di regolare la sfocatura.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.  
Se si sposta il puntatore del mouse lungo l'asse *x*, si modifica la proprietà `blurX` del filtro sfocatura. Se si sposta il puntatore del mouse lungo l'asse *y*, si modifica la proprietà `blurY` del filtro sfocatura. Più il puntatore del mouse viene avvicinato all'angolo superiore sinistro dello stage, meno sfocato sarà il clip filmato.

# Animazione di un filtro mediante ActionScript

È possibile animare filtri in fase di runtime mediante ActionScript, tra cui la classe Tween. Ciò consente di applicare effetti animati interessanti alle applicazioni Flash.

Nell'esempio seguente, viene illustrato come combinare il BlurFilter con la classe Tween per creare una sfocatura animata che consenta di modificare il filtro Sfocatura in base a un valore compreso tra 0 e 10 in fase di runtime.

## Per animare sfocature mediante la classe Tween:

1. Creare un nuovo documento Flash e salvarlo come **animatedfilter.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.filters.BlurFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);

var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
 var myTween:Tween = new Tween(target_mc, "blur", Strong.easeInOut, 0,
 20, 3, true);
 myTween.onMotionChanged = function() {
 target_mc._parent.filters = [new BlurFilter(target_mc.blur,
 target_mc.blur, 1)];
 };
 myTween.onMotionFinished = function() {
 myTween.yoyo();
 }
};
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 holder_mc.img_mc);
```

Il codice precedente viene suddiviso in tre sezioni. La prima sezione consente di importare le classi e i pacchetti richiesti. La seconda sezione consente di creare un clip filmato nidificato per caricare un'immagine e applicare filtri al clip filmato holder. La sezione finale del codice consente di creare una nuova istanza MovieClipLoader e un listener del loader del clip filmato. L'oggetto listener consente di definire la funzione di un gestore di eventi singolo, `onLoadInit`, il quale viene avviato in seguito al caricamento dell'immagine ed è disponibile sullo stage. Non appena l'immagine viene riposizionata al centro dello stage, viene creato un nuovo oggetto Tween che consente di animare il clip filmato e di applicare un filtro sfocatura di 0 e 10.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

## Uso del metodo clone()

Il metodo `clone()` incluso in tutte le classi dei filtri restituisce una nuova istanza del filtro con tutte le proprietà uguali all'istanza del filtro originale. Quando si eseguono operazioni con un filtro, bisogna farne una copia. Per fare ciò, è necessario duplicare il filtro mediante il metodo `clone()`. Se non si duplica un filtro mediante il metodo `clone`, Flash crea soltanto un riferimento al filtro originale. Se Flash crea un riferimento al filtro originale, le modifiche apportate al filtro duplicato modificano anche l'oggetto del filtro originale.

La procedura seguente consente di creare una nuova istanza di un `DropShadowFilter` (`greenDropShadow`), di chiamare il metodo `clone()` per duplicare il filtro ombra esterna verde e di salvare un nuovo filtro denominato `redDropShadow`. Il filtro clonato consente di impostare un nuovo colore d'ombra esterna e i due filtri vengono applicati all'istanza di clip filmato `flower_mc` sullo stage.

### Per utilizzare il metodo `clone`:

1. Creare un nuovo documento Flash e denominarlo `clone.fla`.
2. Creare un clip filmato sullo stage.
3. Selezionare l'istanza di clip filmato e digitare `flower_mc` nella casella di testo Nome istanza della finestra di ispezione Proprietà.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
import flash.filters.DropShadowFilter;
var greenDropShadow:DropShadowFilter = new DropShadowFilter();
greenDropShadow.color = 0x00FF00; // verde
var redDropShadow:DropShadowFilter = greenDropShadow.clone();
redDropShadow.color = 0xFF0000; // rosso
flower_mc.filters = [greenDropShadow, redDropShadow];
```

Il codice precedente consente di creare un nuova istanza del filtro ombra esterna denominata `greenDropShadow`. L'oggetto ombra esterna verde viene duplicato mediante il metodo `DropShadowFilter.clone()`, il quale crea un nuovo oggetto del filtro denominato `redDropShadow`. I filtri ombra esterna verde e ombra esterna rossa vengono applicati all'istanza di clip filmato `flower_mc` sullo stage. Se non si chiama il metodo `clone()`, i due filtri ombra esterna vengono visualizzati in rosso. Questo avviene perché la proprietà `redDropShadow.color` modifica sia l'ombra esterna rossa che quella verde, in quanto la prima contiene un riferimento all'ombra esterna verde.

**5.** Selezionare Controllo > Prova filmato per provare il documento Flash.

Il filtro viene duplicato e clonato ed entrambi i filtri vengono applicati all'istanza `flower_mc`.

Per ulteriori informazioni sul metodo `clone()`, consultare metodo `clone` (`DropShadowFilter.clone`) nella *Guida di riferimento di ActionScript 2.0*. Per informazioni correlate, è anche possibile vedere il metodo `clone()` di tutte le classi dei filtri.

## Creazione di bitmap mediante la classe `BitmapData`

La classe `BitmapData` consente di creare immagini bitmap trasparenti e opache con dimensioni arbitrarie e di manipolarle in vari modi in fase di runtime. Quando si modifica direttamente un oggetto `BitmapData` con ActionScript, è possibile creare immagini molto complesse senza la necessità di ridisegnare costantemente il contenuto di ogni fotogramma partendo dai dati vettoriali in Flash Player. I metodi della classe `BitmapData` supportano una vasta gamma di effetti non disponibili nella scheda Filtri nell'area di lavoro di Flash.

Un oggetto `BitmapData` contiene un array di dati pixel. Questi dati possono rappresentare un'immagine bitmap completamente opaca o trasparente contenente dati per il canale alfa. Entrambi i tipi di oggetto `BitmapData` sono memorizzati sotto forma di buffer di interi a 32 bit. Ogni intero a 32 bit determina le proprietà di un singolo pixel nell'immagine bitmap. Ogni intero a 32 bit è una combinazione di quattro valori *canale* a 8 bit (da zero a 255) che descrivono i valori per la trasparenza alfa e per il rosso, il verde e il blu (ARGB) del pixel.

Per informazioni sulle operazioni con i pacchetti, consultare “[Operazioni con i pacchetti dei filtri](#)” a pagina 547.

È possibile trovare un file di origine di esempio che utilizza la classe `BitmapData` per manipolare un'immagine. Individuare il file denominato `BitmapData.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\BitmapData*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/BitmapData*.

Nella procedura seguente, viene caricata un'immagine JPEG nello stage in modo dinamico e utilizzata la classe `BitmapData` per creare un effetto di disturbo, simile al rumore di staticità causato da un televisore. L'effetto di disturbo viene ridisegnato mediante un motivo casuale ogni 100 millisecondi (1/10 di secondo). Se si sposta il puntatore del mouse lungo l'asse *x* e l'asse *y*, si influisce sulla quantità di staticità disegnata ad ogni intervallo.

**Per creare un effetto di disturbo mediante la classe `BitmapData`:**

1. Creare un nuovo documento Flash e salvarlo come `noise.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import flash.display.BitmapData;
this.createTextField("status_txt", 90, 0, 0, 100, 20);
status_txt.selectable = false;
status_txt.background = 0xFFFFFFFF;
status_txt.autoSize = "left";
function onMouseMove() {
 status_txt._x = _xmouse;
 status_txt._y = _ymouse-20;
 updateAfterEvent();
}
this.createEmptyMovieClip("img_mc", 10);
img_mc.loadMovie("http://www.helppexamples.com/flash/images/image1.jpg");
var noiseBmp:BitmapData = new BitmapData(Stage.width, Stage.height,
 true);
this.attachBitmap(noiseBmp, 20);
setInterval(updateNoise, 100);
var grayScale:Boolean = true;
function updateNoise():Void {
 var low:Number = 30 * _xmouse / Stage.width;
 var high:Number = 200 * _ymouse / Stage.height;
 status_txt.text = "low:" + Math.round(low) + ", high:" +
 Math.round(high);
 noiseBmp.noise(Math.round(Math.random() * 100000), low, high, 8,
 true);
}
```

Questo codice consente di creare un campo di testo col nome di istanza `status_txt`, il quale risponde al puntatore del mouse e visualizza i valori correnti dei parametri `high` e `low` del metodo `noise()`. La funzione `setInterval()` consente di modificare l'effetto di disturbo, il quale viene aggiornato ogni 100 millisecondi (1/10 di secondo), chiamando costantemente la funzione `updateNoise()`. I parametri `high` e `low` del metodo `noise()` vengono determinati calcolando la posizione corrente del puntatore sullo stage.

**3. Selezionare Controllo > Prova filmato per provare il documento.**

Se si sposta il puntatore del mouse lungo l'asse *x* si influisce sul parametro `low`, se si sposta il puntatore del mouse lungo l'asse *y* si influisce sul parametro `high`.

La classe `BitmapData` consente anche di distorcere l'immagine caricata in modo dinamico mediante la combinazione dell'effetto del metodo `perlinNoise()` e del filtro mappa di spostamento. Nella procedura seguente, viene illustrato quanto detto.

**Per applicare un filtro mappa di spostamento a un'immagine:**

1. Creare un nuovo documento Flash e salvarlo come `displacement.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Importa classi.
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
// Crea un clip e un clip nidificato.
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
// Carica JPEG.
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
 image4.jpg", shapeClip.holderClip);
// Crea l'istanza BitmapData.
var perlinBmp:BitmapData = new BitmapData(Stage.width, Stage.height);
perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
 Math.round(Math.random() * 100000), false, true, 1, false);
// Crea e applica il filtro mappa di spostamento.
var displacementMap:DisplacementMapFilter = new
 DisplacementMapFilter(perlinBmp, new Point(0, 0), 1, 1, 100, 100,
 "color", 1);
shapeClip.filters = [displacementMap];
// Crea e applica un listener.
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
 Math.round(Math.random() * 100000), false, true, 1, false);
 shapeClip.filters = [displacementMap];
}
Mouse.addListener(mouseListener);
```

L'esempio di codice seguente si divide in cinque sezioni logiche. Nella prima sezione, vengono importate le classi necessarie dell'esempio. Il secondo blocco consente di creare un clip filmato nidificato e di caricare un'immagine JPEG da un server remoto. La terza parte consente di creare una nuova istanza BitmapData denominata `perlinBmp`, avente le stesse dimensioni dello stage. L'istanza `perlinBmp` contiene i risultati dell'effetto di disturbo Perlin, utilizzato in seguito come parametro del filtro mappa di spostamento. Il quarto blocco consente di applicare l'effetto del filtro mappa di spostamento all'immagine caricata in modo dinamico creata precedentemente. Il quinto e ultimo blocco consente di creare un listener del mouse, il quale rigenera il disturbo Perlin utilizzato dal filtro mappa di spostamento ogni qualvolta viene spostato il puntatore del mouse.

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

## Informazioni sui metodi di fusione

È possibile applicare i metodi di fusione a oggetti del clip filmato mediante l'area di lavoro di Flash (Flash Professional 8) o ActionScript (Flash Basic 8 e Flash Professional 8). In fase di runtime, vengono riuniti più simboli grafici in un'unica forma. Per questo motivo, non è possibile applicare diversi metodi di fusione a simboli grafici diversi.

Per ulteriori informazioni sull'applicazione dei metodi di fusione mediante ActionScript, consultare [“Applicazione dei metodi di fusione” a pagina 588](#).

I metodi di fusione consentono di combinare i colori di un'immagine (l'immagine di base) con quelli di un'altra immagine (l'immagine di fusione) per produrre una terza immagine. Ogni valore di pixel di un'immagine viene elaborato con il valore di pixel corrispondente dell'altra immagine per produrre un valore di pixel per la stessa posizione all'interno del risultato.

La proprietà `MovieClip.blendMode` supporta i metodi di fusione seguenti:

**add** Utilizzato solitamente per creare una dissolvenza con effetto di schiarimento animato tra due immagini.

**alpha** Utilizzato solitamente per applicare la trasparenza del primo piano sullo sfondo.

**darken** Utilizzato solitamente per la sovrapposizione del tipo.

**difference** Utilizzato solitamente per creare colori più vivaci.

**erase** Utilizzato solitamente per *ritagliare* (cancellare) parte dello sfondo mediante l'alfa di primo piano.

**hardlight** Utilizzato solitamente per creare effetti di ombreggiatura.

**invert** utilizzato per invertire lo sfondo.

- layer** Utilizzato per imporre la creazione di un buffer temporaneo per la precomposizione di un clip filmato particolare.
- lighten** Utilizzato solitamente per la sovrapposizione del tipo.
- multiply** Utilizzato solitamente per creare ombre ed effetti di profondità.
- normal** Utilizzato per specificare che i valori dei pixel dell'immagine a cui è stata applicata la fusione hanno la priorità su quelli dell'immagine di base.
- overlay** Utilizzato solitamente per creare effetti di ombreggiatura.
- screen** Utilizzato solitamente per evidenziazioni ed effetti di riflesso lente.
- subtract** Utilizzato solitamente per creare una dissolvenza con effetto di oscuramento animato tra due immagini.

## Applicazione dei metodi di fusione

La procedura seguente consente di caricare un'immagine dinamica e di applicare vari metodi di fusione all'immagine selezionandone uno nella casella combinata sullo stage.

### Per applicare vari metodi di fusione a un'immagine:

1. Creare un nuovo documento Flash e salvarlo come **blendmodes.fla**.
2. Trascinare un componente ComboBox nello stage e assegnare ad esso il nome di istanza **blendMode\_cb**.
3. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```

var blendMode_dp:Array = new Array();
blendMode_dp.push({data:"add", label:"add"});
blendMode_dp.push({data:"alpha", label:"alpha"});
blendMode_dp.push({data:"darken", label:"darken"});
blendMode_dp.push({data:"difference", label:"difference"});
blendMode_dp.push({data:"erase", label:"erase"});
blendMode_dp.push({data:"hardlight", label:"hardlight"});
blendMode_dp.push({data:"invert", label:"invert"});
blendMode_dp.push({data:"layer", label:"layer"});
blendMode_dp.push({data:"lighten", label:"lighten"});
blendMode_dp.push({data:"multiply", label:"multiply"});
blendMode_dp.push({data:"normal", label:"normal"});
blendMode_dp.push({data:"overlay", label:"overlay"});
blendMode_dp.push({data:"screen", label:"screen"});
blendMode_dp.push({data:"subtract", label:"subtract"});
blendMode_cb.dataProvider = blendMode_dp;

var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
 var blendModeClip:MovieClip =
 target_mc.createEmptyMovieClip("blendModeType_mc", 20);
}

```

```

with (blendModeClip) {
 beginFill(0x999999);
 moveTo(0, 0);
 lineTo(target_mc._width / 2, 0);
 lineTo(target_mc._width / 2, target_mc._height);
 lineTo(0, target_mc._height);
 lineTo(0, 0);
 endFill();
}
target_mc._x = (Stage.width - target_mc._width) / 2;
target_mc._y = (Stage.height - target_mc._height) / 2;
blendModeClip.blendMode = blendMode_cb.value;
};

this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 img_mc);

function cbListener(eventObj:Object):Void {
 img_mc.blendModeType_mc.blendMode = eventObj.target.value;
}
blendMode_cb.addEventListener("change", cbListener);

```

Questo codice ActionScript consente di inserire i metodi di fusione nella casella combinata. In questo modo, è possibile visualizzare ciascun effetto sull'immagine caricata in modo dinamico. Viene creato un oggetto listener utilizzato con un'istanza MovieClipLoader. L'oggetto listener consente di definire la funzione di un gestore di eventi singolo, onLoadInit, il quale viene chiamato quando l'immagine viene caricata completamente e inizializzata mediante Flash. Il listener dell'evento consente di creare un nuovo clip filmato denominato blendModeType\_mc e di disegnare una forma rettangolare mediante l'API sulla metà sinistra dell'immagine. Il metodo di fusione dell'istanza ComboBox correntemente selezionata viene quindi applicato al clip filmato blendModeType\_mc.

Il codice rimanente consente di impostare l'istanza MovieClipLoader, responsabile del caricamento dell'immagine specificata nel clip filmato sullo stage. Infine, viene definito un listener dell'istanza Combo Box blendMode\_cb, il quale viene applicato al metodo di fusione ogni qualvolta si seleziona un nuovo elemento dall'istanza ComboBox.

4. Selezionare Controllo > Prova filmato per provare il documento.

# Informazioni sull'ordine delle operazioni

L'elenco seguente costituisce l'ordine delle operazioni in cui un array dei filtri, i metodi di fusione, gli elementi di trasformazione del colore e i livelli di maschera vengono allegati o eseguiti in relazione a un'istanza di clip filmato.

1. La bitmap di un clip filmato viene aggiornata nel contenuto vettoriale (la proprietà `cacheAsBitmap` è impostata su `true`).
2. Se si utilizza il metodo `setMask()` e la maschera dispone di una cache delle bitmap, Flash esegue una fusione alfa tra le due immagini.
3. Vengono quindi applicati i filtri (sfocatura, ombra esterna, bagliore e così via).
4. Se si utilizza la classe `ColorTransform`, l'operazione di trasformazione del colore viene eseguita e memorizzata nella cache come bitmap risultante.
5. Se si applica un metodo di fusione, la fusione viene eseguita mediante un renderer dei vettori.
6. Se si applicano livelli di mascheramento esterni, i livelli eseguono il mascheramento (mediante un renderer dei vettori).

## Disegno con ActionScript

Per disegnare linee e riempimenti sullo stage, è possibile utilizzare i metodi della classe `MovieClip`. In questo modo si possono creare strumenti di disegno per gli utenti e disegnare forme nel file SWF in risposta a eventi. I metodi di disegno della classe `MovieClip` sono i seguenti:

- `beginFill()`
- `beginGradientFill()`
- `clear()`
- `curveTo()`
- `endFill()`
- `lineTo()`
- `lineStyle()`
- `moveTo()`

I metodi di disegno possono essere utilizzati con qualunque clip filmato. Tuttavia, se vengono utilizzati con un clip filmato realizzato in modalità di creazione, i metodi di disegno vengono eseguiti prima che il clip venga disegnato. In altre parole, il contenuto creato in modalità di creazione viene tracciato sopra al contenuto tracciato con i metodi di disegno.

È possibile utilizzare i clip filmato con metodi di disegno come le maschere; tuttavia, come accade con tutte le maschere dei clip filmato, i tratti vengono ignorati.

Per ulteriori informazioni sul disegno mediante ActionScript, consultare i seguenti argomenti:

- “[Uso dei metodi di disegno per creare linee, curve e forme](#)” a pagina 591
- “[Disegno di forme specifiche](#)” a pagina 593
- “[Uso di riempimenti con gradiente complessi](#)” a pagina 597
- “[Uso degli stili di linea](#)” a pagina 598
- “[Uso dei metodi dell'API di disegno e dell'animazione con script](#)” a pagina 604

È possibile trovare un file di origine di esempio, drawingapi.fla, nella cartella Samples sul disco rigido, in cui viene descritto come utilizzare l'API di disegno in un'applicazione Flash.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DrawingAPI*.

## Uso dei metodi di disegno per creare linee, curve e forme

In fase di runtime, è possibile creare forme sullo stage mediante l'API di disegno di Flash. È possibile utilizzare queste forme per mascherare il contenuto in modo dinamico, applicarvi dei filtri o animarle intorno allo stage. È anche possibile creare vari strumenti di disegno mediante l'API di disegno, il quale consente di disegnare forme sul file SWF mediante il mouse o la tastiera.

### Per disegnare una linea:

1. Creare un nuovo documento Flash e salvarlo come **line.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("line_mc", 10);
line_mc.lineStyle(1, 0x000000, 100);
line_mc.moveTo(0, 0);
line_mc.lineTo(200, 100);
line_mc._x = 100;
line_mc._y = 100;
```

Questo codice consente di disegnare una linea compresa tra 0,0 sullo stage e 200,100. Le coordinate *\_x* e *\_y* della linea vengono quindi modificate per riposizionare la linea a 100,100 sullo stage.
3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Per disegnare una forma più complessa, continuare a chiamare il metodo `MovieClip.lineTo()` e disegnare un rettangolo, un quadrato o un ovale, come illustrato nella procedura seguente.

#### Per disegnare una curva:

1. Creare un nuovo documento Flash e salvarlo come **curve.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
 lineStyle(4, 0x000000, 100);
 beginFill(0xFF0000);
 moveTo(200, 300);
 curveTo(300, 300, 300, 200);
 curveTo(300, 100, 200, 100);
 curveTo(100, 100, 100, 200);
 curveTo(100, 300, 200, 300);
 endFill();
}
```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento Flash.

In questo codice, viene disegnato un cerchio sullo stage mediante l'API di disegno. La forma del cerchio utilizza solo quattro chiamate al metodo `MovieClip.curveTo()`. Pertanto, è possibile che venga visualizzata una forma un po' distorta. Un esempio ulteriore della creazione di un cerchio mediante l'API di disegno è contenuto nella procedura relativa alla creazione di un cerchio in [“Disegno di forme specifiche”](#) a pagina 593 per il codice che utilizza otto chiamate al metodo `MovieClip.curveTo()` per disegnare un cerchio più realistico.

#### Per disegnare un triangolo:

1. Creare un nuovo documento Flash e salvarlo come **triangle.fla**.

2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("triangle_mc", 1);
```

Questo codice utilizza il metodo `MovieClip.createEmptyMovieClip()` per creare un clip filmato vuoto sullo stage. Il nuovo clip filmato deriva da uno esistente (in questo caso, la linea temporale principale).

- 3.** Aggiungere il seguente codice ActionScript al fotogramma 1 della linea temporale, di seguito al codice aggiunto al punto precedente:

```
with (triangle_mc) {
 lineStyle(5, 0xFF00FF, 100);
 moveTo(200, 200);
 lineTo(300, 300);
 lineTo(100, 300);
 lineTo(200, 200);
}
```

In questo codice, il clip filmato vuoto (`triangle_mc`) chiama i metodi di disegno. Questo codice consente di disegnare un triangolo con linee viola di 5 pixel senza riempimento.

- 4.** Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento Flash.

Per informazioni dettagliate su questi metodi, consultare le relative sezioni in `MovieClip` nella *Guida di riferimento di ActionScript 2.0*.

È possibile trovare un file di origine di esempio, `drawingapi.fla`, nella cartella Samples sul disco rigido, in cui viene descritto come utilizzare l'API di disegno in un'applicazione Flash.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DrawingAPI*.

## Disegno di forme specifiche

In questa sezione, viene descritto come creare metodi più flessibili per disegnare forme più avanzate, come rettangoli arrotondati e cerchi.

### Per creare un rettangolo:

1. Creare un nuovo documento Flash e salvarlo come `rect.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRectangle(rectangle_mc, 240, 180, 0x99FF00, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number,
 boxHeight:Number, fillColor:Number, fillAlpha:Number):Void {
 with (target_mc) {
 beginFill(fillColor, fillAlpha);
 moveTo(0, 0);
 lineTo(boxWidth, 0);
 lineTo(boxWidth, boxHeight);
 }
```

```

 lineTo(0, boxHeight);
 lineTo(0, 0);
 endFill();
 }
}

```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento Flash.

Flash consente di disegnare un rettangolo verde semplice sullo stage e di posizionarlo a 100,100. Per modificarne le dimensioni, il colore di riempimento o la trasparenza, è possibile modificare i valori nella chiamata al metodo `drawRectangle()` anziché i contenuti del metodo `MovieClip.beginFill()`.

È anche possibile creare un rettangolo con angoli arrotondati mediante l'API di disegno, come descritto nella procedura seguente.

#### **Per creare un rettangolo arrotondato:**

1. Creare un nuovo documento Flash e salvarlo come **roundrect.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```

this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRoundedRectangle(rectangle_mc, 240, 180, 20, 0x99FF00, 100);
function drawRoundedRectangle(target_mc:MovieClip, boxWidth:Number,
 boxHeight:Number, cornerRadius:Number, fillColor:Number,
 fillAlpha:Number):Void {
 with (target_mc) {
 beginFill(fillColor, fillAlpha);
 moveTo(cornerRadius, 0);
 lineTo(boxWidth - cornerRadius, 0);
 curveTo(boxWidth, 0, boxWidth, cornerRadius);
 lineTo(boxWidth, cornerRadius);
 lineTo(boxWidth, boxHeight - cornerRadius);
 curveTo(boxWidth, boxHeight, boxWidth - cornerRadius, boxHeight);
 lineTo(boxWidth - cornerRadius, boxHeight);
 lineTo(cornerRadius, boxHeight);
 curveTo(0, boxHeight, 0, boxHeight - cornerRadius);
 lineTo(0, boxHeight - cornerRadius);
 lineTo(0, cornerRadius);
 curveTo(0, 0, cornerRadius, 0);
 lineTo(cornerRadius, 0);
 endFill();
 }
}

```

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento.

Viene visualizzato un rettangolo verde sullo stage di 240 pixel di larghezza e di 180 pixel di altezza con angoli arrotondati di 20-pixel. È possibile creare istanze multiple di rettangoli arrotondati realizzando clip filmato nuovi mediante

`MovieClip.createEmptyMovieClip()` e chiamando la funzione personalizzata `drawRoundedRectangle()`.

È possibile creare un cerchio perfetto mediante l'API di disegno, come illustrato nella procedura seguente.

#### Per creare un cerchio:

1. Creare un nuovo documento Flash e salvarlo come **circle2.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("circle_mc", 10);
circle_mc._x = 100;
circle_mc._y = 100;
drawCircle(circle_mc, 100, 0x99FF00, 100);

function drawCircle(target_mc:MovieClip, radius:Number,
 fillColor:Number, fillAlpha:Number):Void {
 var x:Number = radius;
 var y:Number = radius;
 with (target_mc) {
 beginFill(fillColor, fillAlpha);
 moveTo(x + radius, y);
 curveTo(radius + x, Math.tan(Math.PI / 8) * radius + y,
 Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius + y);
 curveTo(Math.tan(Math.PI / 8) * radius + x, radius + y, x, radius + y);
 curveTo(-Math.tan(Math.PI / 8) * radius + x, radius + y,
 -Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius + y);
 curveTo(-radius + x, Math.tan(Math.PI / 8) * radius + y, -radius + x, y);
 curveTo(-radius + x, -Math.tan(Math.PI / 8) * radius + y,
 -Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius + y);
 curveTo(-Math.tan(Math.PI / 8) * radius + x, -radius + y, x, -radius + y);
 curveTo(Math.tan(Math.PI / 8) * radius + x, -radius + y,
 Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius + y);
 curveTo(radius + x, -Math.tan(Math.PI / 8) * radius + y, radius + x, y);
 endFill();
 }
}
```

**3.** Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

Questo codice consente di creare un cerchio più complesso e realistico di quello mostrato nell'esempio precedente. Invece di utilizzare soltanto le quattro chiamate del metodo `curveTo()`, questo esempio utilizza le otto del metodo `curveTo()`, conferendo al cerchio un aspetto più arrotondato.

È possibile creare un triangolo mediante l'API di disegno, come illustrato nella procedura seguente.

**Per creare un triangolo personalizzato:**

**1.** Creare un nuovo documento Flash e salvarlo come **fancytriangle.fla**.

**2.** Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("triangle_mc", 10);
triangle_mc._x = 100;
triangle_mc._y = 100;
drawTriangle(triangle_mc, 100, 0x99FF00, 100);

function drawTriangle(target_mc:MovieClip, sideLength:Number,
 fillColor:Number, fillAlpha:Number):Void {
 var tHeight:Number = sideLength * Math.sqrt(3) / 2;
 with (target_mc) {
 beginFill(fillColor, fillAlpha);
 moveTo(sideLength / 2, 0);
 lineTo(sideLength, tHeight);
 lineTo(0, tHeight);
 lineTo(sideLength / 2, 0);
 endFill();
 }
}
```

L'API di disegno consente di disegnare un triangolo equilatero sullo stage e di riempirlo con il colore di riempimento e la quantità di alfa (trasparenza) specificati.

**3.** Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento.

È possibile trovare un file di origine di esempio, **drawingapi.fla**, nella cartella Samples sul disco rigido, in cui viene descritto come utilizzare l'API di disegno in un'applicazione Flash.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/DrawingAPI*.

## Uso di riempimenti con gradiente complessi

L'API di disegno di Flash supporta sia i riempimenti con gradiente che quelli uniformi. La procedura seguente consente di creare un nuovo clip filmato sullo stage, di disegnare un quadrato mediante l'API di disegno e di riempirlo con un gradiente radiale rosso e blu.

### Per creare un gradiente complesso:

1. Creare un nuovo documento Flash e salvarlo come **radialgradient.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("gradient_mc", 10);
var fillType:String = "radial";
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
var matrix:Object = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200,
 i:1};
var spreadMethod:String = "reflect";
var interpolationMethod:String = "linearRGB";
var focalPointRatio:Number = 0.9;
with (gradient_mc) {
 beginGradientFill(fillType, colors, alphas, ratios, matrix,
 spreadMethod, interpolationMethod, focalPointRatio);
 moveTo(100, 100);
 lineTo(100, 300);
 lineTo(300, 300);
 lineTo(300, 100);
 lineTo(100, 100);
 endFill();
}
```

Nel codice ActionScript precedente, viene creato un quadrato sullo stage mediante l'API di disegno e chiamato il metodo `beginGradientFill()` per riempire il quadrato con un gradiente circolare rosso e blu.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file di Flash.

## Uso degli stili di linea

L'API di disegno di Flash consente di specificare lo stile di una linea utilizzato dall'applicazione per le chiamate successive a `MovieClip.lineTo()` e `MovieClip.curveTo()` fino alla chiamata di `MovieClip.lineStyle()` con parametri diversi, che sono i seguenti:

```
lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean,
noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number)
```

È possibile chiamare `MovieClip.lineStyle()` mentre si disegna un percorso per specificare stili diversi per segmenti di linea diversi all'interno di un percorso.

Per maggiori informazioni sull'impostazione di stili di linee mediante ActionScript, consultare le sezioni seguenti:

- “[Impostazione degli stili di tratti ed estremità](#)” a pagina 598
- “[Impostazione dei parametri degli stili di linea](#)” a pagina 599

### Impostazione degli stili di tratti ed estremità

Flash 8 include diversi miglioramenti legati al disegno delle linee. I parametri delle linee introdotti in Flash Player 8 sono: `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` e `miterLimit`.

Nella procedura seguente, viene illustrata la differenza fra i tre nuovi stili di estremità in Flash Player 8: `none`, `round` e `square`.

#### Per impostare stili di estremità mediante ActionScript:

1. Creare un nuovo documento Flash e salvarlo come `capstyle.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
// Imposta il clip filmato della griglia.
this.createEmptyMovieClip("grid_mc", 50);
grid_mc.lineStyle(0, 0x999999, 100);
grid_mc.moveTo(50, 0);
grid_mc.lineTo(50, Stage.height);
grid_mc.moveTo(250, 0);
grid_mc.lineTo(250, Stage.height);
// linea 1 (capsStyle: round)
this.createEmptyMovieClip("line1_mc", 10);
with (line1_mc) {
 createTextField("label_txt", 1, 5, 10, 100, 20);
 label_txt.text = "round";
 lineStyle(20, 0x99FF00, 100, true, "none", "round", "miter", 0.8);
 moveTo(0, 0);
 lineTo(200, 0);
 _x = 50;
 _y = 50;
```

```

}
// linea 2 (capsStyle: square)
this.createEmptyMovieClip("line2_mc", 20);
with (line2_mc) {
 createTextField("label_txt", 1, 5, 10, 100, 20);
 label_txt.text = "square";
 lineStyle(20, 0x99FF00, 100, true, "none", "square", "miter", 0.8);
 moveTo(0, 0);
 lineTo(200, 0);
 _x = 50;
 _y = 150;
}
// linea 3 (capsStyle: none)
this.createEmptyMovieClip("line3_mc", 30);
with (line3_mc) {
 createTextField("label_txt", 1, 5, 10, 100, 20);
 label_txt.text = "none";
 lineStyle(20, 0x99FF00, 100, true, "none", "none", "miter", 0.8);
 moveTo(0, 0);
 lineTo(200, 0);
 _x = 50;
 _y = 250;
}

```

Il codice precedente consente di creare quattro clip filmato in modo dinamico e di disegnare una serie di linee sullo stage mediante l'API di disegno. Il primo clip filmato contiene due linee verticali, una di 50 pixel e l'altra di 250 pixel sull'asse x. Nei tre clip filmato successivi, viene disegnata un linea verde sullo stage e il capsStyle relativo viene impostato su round, square o none.

### 3. Selezionare Controllo > Prova filmato per provare il documento.

Gli stili di estremità vengono visualizzati sullo stage in fase di runtime.

## Impostazione dei parametri degli stili di linea

È possibile impostare i parametri degli stili di linea in modo da modificare l'aspetto dei tratti. È possibile utilizzare parametri per modificare lo spessore, il colore, l'alfa, la scala e altri attributi dello stile di linea.

### Impostazione dello spessore della linea

Il parametro thickness del metodo MovieClip.lineStyle() consente di specificare lo spessore della linea disegnata espresso in punti come numero. È possibile disegnare una linea di spessore compreso tra 0 e 255 punti di ampiezza, anche se l'impostazione di uno spessore pari a 0 crea il cosiddetto *spessore sottilissimo*. Il tratto è sempre di 1 pixel, indipendentemente dal fatto che il file SWF venga rimpicciolito o ridimensionato.

Nella procedura seguente, viene illustrata la differenza tra una linea con spessore standard di 1 pixel e una linea con spessore sottilissimo.

#### Per creare un tratto sottilissimo:

1. Creare un nuovo documento Flash e salvarlo come **hairline.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
this.createEmptyMovieClip("drawing_mc", 10);
// Crea una linea rossa e con spessore sottilissimo
drawing_mc.lineStyle(0, 0xFF0000, 100);
drawing_mc.moveTo(0, 0);
drawing_mc.lineTo(200, 0);
drawing_mc.lineTo(200, 100);
// Crea una linea blu con spessore di 1 pixel
drawing_mc.lineStyle(1, 0x0000FF, 100);
drawing_mc.lineTo(0, 100);
drawing_mc.lineTo(0, 0);
drawing_mc._x = 100;
drawing_mc._y = 100;
```

Nel codice precedente, vengono disegnate due linee sullo stage mediante l'API di disegno. La prima linea è rossa e ha uno spessore pari a 0 (spessore sottilissimo), mentre la seconda è blu e ha uno spessore di 1 pixel.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF. Inizialmente, sia la linea rossa che quella blu hanno lo stesso aspetto. Se si fa clic col tasto destro del mouse sul file SWF e si seleziona Ingrandisci nel menu di scelta rapida, la linea rossa viene visualizzata sempre come una linea di 1 pixel, mentre quella blu diventa sempre più grande ogni qualvolta si ingrandisce il file SWF.

#### Impostazione del colore della linea (rgb)

Il secondo parametro nel metodo `lineStyle()`, `rgb`, consente di controllare il colore del segmento della linea espresso in numeri. Per impostazione predefinita, Flash disegna linee nere (#000000), anche se è possibile specificare colori diversi impostando un nuovo valore esadecimale del colore mediante la sintassi `0xRRGGBB`. Nella sintassi seguente, RR costituisce il valore rosso (tra 00 e FF), GG il valore verde (00 a FF) e BB quello blu (00 a FF).

Ad esempio, si rappresenta una linea rossa con `0xFF0000`, una linea verde con `0x00FF00`, una linea blu con `0x0000FF`, una linea viola con `0xFF00FF` (rosso e blu), una linea bianca con `#FFFFFF`, una linea grigia con `#999999` e così via.

### Impostazione dell'alfa della linea

Il terzo parametro nel metodo `lineStyle()`, `alpha`, consente di controllare il livello di trasparenza (alfa) della linea. La trasparenza è costituita da un valore numerico compreso tra 0 e 100, dove 0 rappresenta una linea completamente trasparente e 100 una linea completamente opaca (visibile).

### Impostazione dell'approssimazione dei pixel della linea (pixelHinting)

L'approssimazione dei pixel del parametro relativo ai tratti, `pixelHinting`, consiste nell'impostazione dei punti di ancoraggio di linee e curve su pixel pieni. I tratti sono su pixel pieni per tutti gli spessori dei tratti, ossia non è mai possibile visualizzare una linea orizzontale o verticale sfocata. È necessario impostare il parametro `pixelHinting` sul valore booleano (`true` o `false`).

### Impostazione della modifica in scala della linea (noScale)

È necessario impostare il parametro `noScale` mediante un valore di tipo stringa che consenta di specificare una modalità di modifica in scala della linea. È anche possibile utilizzare un tratto non modificabile in scala in modo orizzontale o verticale, modificare in scala la linea (normale) o non fare ricorso a questo procedimento.

SUGGERIMENTO

È consigliabile attivare la modifica in scala per gli elementi dell'interfaccia utente ingranditi dagli utenti, ma non se un clip filmato viene modificato in scala soltanto in verticale o in orizzontale.

È possibile utilizzare una delle quattro diverse modalità per specificare i casi in cui la modifica in scala deve o non deve essere applicata. I valori possibili della proprietà `noScale` sono i seguenti:

`normal` Applica sempre la modifica in scala dello spessore (*impostazione predefinita*).

`vertical` Non applica la modifica in scala dello spessore se l'oggetto viene modificato in scala verticalmente.

`horizontal` Non applica la modifica in scala dello spessore se l'oggetto viene modificato in scala orizzontalmente.

`none` Non modifica mai in scala lo spessore.

## Impostazione delle estremità della linea (`capsStyle`) e degli spigoli (`jointStyle`)

È possibile impostare tre tipi di stili di estremità per il parametro `capsStyle`:

- `round` (*impostazione predefinita*)
- `square`
- `none`

Nella procedura seguente, vengono illustrate le differenze fra i tre stili di estremità. Quando si prova il file SWF, viene visualizzata una rappresentazione visiva di tutti gli stili di estremità.

### Per impostare stili di estremità diversi:

1. Creare un nuovo documento Flash e salvarlo come `capsstyle2.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
var lineLength:Number = 100;
// Arrotondato
this.createEmptyMovieClip("round_mc", 10);
round_mc.lineStyle(20, 0xFF0000, 100, true, "none", "round");
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc.lineStyle(0, 0x000000);
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc._x = 50;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0, 10,
 lineLength, 20);
lbl.text = "round";
var lineLength:Number = 100;
// Quadrato
this.createEmptyMovieClip("square_mc", 20);
square_mc.lineStyle(20, 0xFF0000, 100, true, "none", "square");
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc.lineStyle(0, 0x000000);
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc._x = 200;
square_mc._y = 50;
var lbl:TextField = square_mc.createTextField("label_txt", 10, 0, 10,
 lineLength, 20);
lbl.text = "square";
// Nessuno
this.createEmptyMovieClip("none_mc", 30);
none_mc.lineStyle(20, 0xFF0000, 100, true, "none", "none");
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
```

```

none_mc.lineStyle(0, 0x000000);
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc._x = 350;
none_mc._y = 50;
var lbl:TextField = none_mc.createTextField("label_txt", 10, 0, 10,
 lineLength, 20);
lbl.text = "none";

```

Il codice precedente consente di disegnare tre linee mediante l'API di disegno, ognuna con un valore diverso per capsStyle.

### 3. Selezionare Controllo > Prova filmato per provare il documento Flash.

È possibile impostare tre tipi di stili di spigoli per il parametro jointStyle:

- round (*impostazione predefinita*)
- miter
- bevel

Nella procedura seguente, vengono dimostrate le differenze fra i tre stili di spigoli.

#### Per impostare stili di spigolo diversi:

1. Creare un nuovo documento Flash e salvarlo come **jointstyles.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```

var lineLength:Number = 100;
// Giunzione ad angolo
this.createEmptyMovieClip("miter_mc", 10);
miter_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "miter",
 25);
miter_mc.moveTo(0, lineLength);
miter_mc.lineTo(lineLength / 2, 0);
miter_mc.lineTo(lineLength, lineLength);
miter_mc.lineTo(0, lineLength);
miter_mc._x = 50;
miter_mc._y = 50;
var lbl:TextField = mitter_mc.createTextField("label_txt", 10, 0,
 lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "miter";
// Arrotondato
this.createEmptyMovieClip("round_mc", 20);
round_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "round");
round_mc.moveTo(0, lineLength);
round_mc.lineTo(lineLength / 2, 0);
round_mc.lineTo(lineLength, lineLength);
round_mc.lineTo(0, lineLength);
round_mc._x = 200;
round_mc._y = 50;

```

```

var lbl:TextField = round_mc.createTextField("label_txt", 10, 0,
 lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "round";
// Smussatura
this.createEmptyMovieClip("bevel_mc", 30);
bevel_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "bevel");
bevel_mc.moveTo(0, lineLength);
bevel_mc.lineTo(lineLength / 2, 0);
bevel_mc.lineTo(lineLength, lineLength);
bevel_mc.lineTo(0, lineLength);
bevel_mc._x = 350;
bevel_mc._y = 50;
var lbl:TextField = bevel_mc.createTextField("label_txt", 10, 0,
 lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "bevel";

```

Flash disegna tre triangoli sullo stage mediante l'API di disegno. Ciascun triangolo dispone di un valore diverso per ogni stile di spigoli.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il documento.

### Impostazione della giunzione ad angolo della linea (miterLimit)

La proprietà `miterLimit` è costituita da un valore numerico che indica il limite in cui la giunzione ad angolo di uno spigolo (vedere [“Impostazione delle estremità della linea \(capsStyle\) e degli spigoli \(jointStyle\)” a pagina 602](#)) viene troncata. Il valore `miterLimit` costituisce il moltiplicatore generale di un tratto. Ad esempio, con un valore di 2,5, `miterLimit` viene troncato ogni 2,5 volte le dimensioni del tratto. L'intervallo di valori valido è compreso tra 0 e 255 (se un valore di `miterLimit` è `undefined`, il valore predefinito è pari a 3). La proprietà `miterLimit` viene utilizzata soltanto se `jointStyle` è impostato `miter`.

## Uso dei metodi dell'API di disegno e dell'animazione con script

È possibile combinare l'API di disegno con le classi `Tween` e `TransitionManager` per creare animazioni di eccellente qualità. Per fare ciò, è sufficiente scrivere una piccola quantità di codice ActionScript.

La procedura seguente consente di caricare un'immagine JPEG e di mascherarla dinamicamente in modo da rivelarla lentamente dopo il suo caricamento mediante l'interpolazione della maschera dell'immagine.

### **Per animare maschere dinamiche:**

1. Creare un nuovo documento Flash e salvarlo come **dynmask.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
 target_mc._visible = false;
 // Centra l'immagine sullo stage
 target_mc._x = (Stage.width - target_mc._width) / 2;
 target_mc._y = (Stage.height - target_mc._height) / 2;
 var maskClip:MovieClip = target_mc.createEmptyMovieClip("mask_mc",
20);
 with (maskClip) {
 // Disegnare una maschera con la stessa dimensione dell'immagine
 caricata.
 beginFill(0xFF00FF, 100);
 moveTo(0, 0);
 lineTo(target_mc._width, 0);
 lineTo(target_mc._width, target_mc._height);
 lineTo(0, target_mc._height);
 lineTo(0, 0);
 endFill();
 }
 target_mc.setMask(maskClip);
 target_mc._visible = true;
 var mask_tween:Object = new Tween(maskClip, "_yscale", Strong.easeOut,
0, 100, 2, true);
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
img_mc);
```

In questo esempio di codice, vengono importate tutte le classi, tra cui la classe Tween, nel pacchetto di andamento. In seguito, viene creato un oggetto del listener di un'istanza MovieClipLoader creata in una sezione successiva del codice. L'oggetto del listener consente di definire il listener di un evento singolo, onLoadInit, che centra l'immagine JPEG caricata in modo dinamico sullo stage. Una volta riposizionata l'immagine con il codice, viene creata una nuova istanza di clip filmato nel clip filmato target\_mc (contenente l'immagine JPEG caricata in modo dinamico). Il codice dell'API di disegno consente di disegnare un rettangolo con le stesse dimensioni dell'immagine JPEG presente nel clip filmato. Il nuovo clip filmato maschera l'immagine JPEG chiamando il metodo MovieClip.setMask(). Una volta disegnata e impostata la maschera, viene utilizzata la classe Tween per animare, provocando il rivelamento rallentato dell'immagine stessa.

3. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

**NOTA**

Per animare `_alpha` nell'esempio precedente anziché `_yscale`, interpolare direttamente `target_mc` invece del clip filmato della maschera.

È possibile trovare un file di origine di esempio, `drawingapi.fla`, nella cartella Samples sul disco rigido, in cui viene descritto come utilizzare l'API di disegno in un'applicazione Flash.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript\DrawingAPI*.

## Nozioni fondamentali sulla modifica in scala e sulle guide porzione

La modifica in scala a 9 porzioni (scala 9) consente di specificare la modifica in scala a livello di stile di componente per i clip filmato. La specifica della modifica in scala a 9 porzioni consente di creare simboli di clip filmato che eseguono una modifica in scala da utilizzare sui componenti dell'interfaccia utente, diversa da quella solitamente applicata agli elementi di grafica e progettazione.

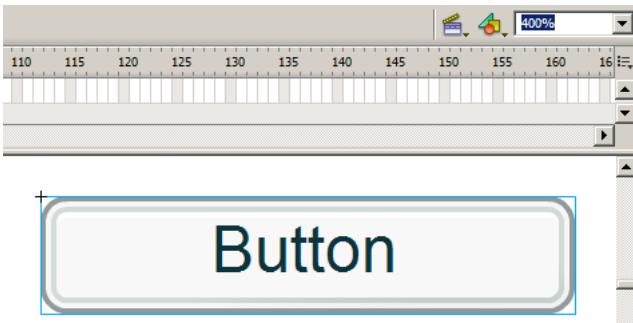
## Nozioni fondamentali sul funzionamento della modifica in scala a 9 porzioni

Per comprendere meglio la modifica in scala a 9 porzioni, è possibile vedere un esempio del suo funzionamento in Flash.

### Per comprendere la modifica in scala in Flash:

1. Creare un nuovo documento Flash e salvarlo come `dynmask.fla`.
2. Trascinare una copia del componente Button sullo stage dal pannello Componenti (Finestra > Componenti).

- 3.** Ingrandire lo stage del 400% mediante lo strumento di riduzione/ingrandimento.



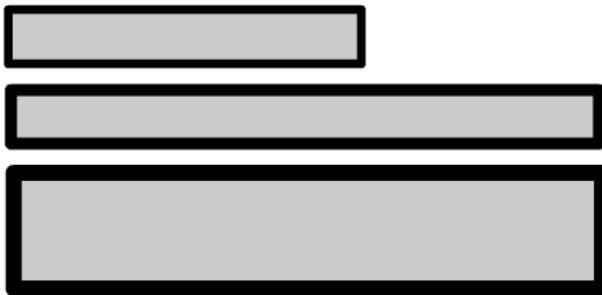
Per impostazione predefinita, l'istanza del componente Button è larga 100 pixel e alta 22 pixel.

- 4.** Ridimensionare l'istanza del componente Button a 200 pixel di larghezza e 44 pixel di altezza mediante la finestra di ispezione Proprietà.



Anche se il componente viene ridimensionato, i bordi del pulsante e l'etichetta di testo non vengono distorti. L'etichetta del pulsante resta centrata e mantiene le dimensioni originali. I componenti della versione 2 di Macromedia Component Architecture non utilizzano la modifica in scala a 9 porzioni Tuttavia, essi sono in grado di gestirla in questa versione per evitare di modificare le dimensioni dei contorni (come mostrato nella figura successiva).

Supporre che l'istanza del pulsante venga suddivisa in 9 pezzi separati o in una griglia 3x3, simile al tastierino numerico di un telefono o di una tastiera. Quando l'istanza del pulsante viene ridimensionata in orizzontale, soltanto i tre segmenti verticali al centro (numeri 2, 5 e 8 su un tastierino numerico) risultano allungati. In questo modo, il contenuto non appare distorto. Se l'istanza del pulsante viene ridimensionata in verticale, vengono ridimensionati soltanto i tre segmenti orizzontali al centro (numeri 4, 5 e 6 su un tastierino numerico). I quattro angoli della griglia non vengono modificati in scala, consentendo al componente di ingrandirsi senza risultare allungato (vedere le immagini seguenti).



**SUGGERIMENTO**

I tratti vengono creati dai bordi in seguito alla trasformazione derivante dalla modifica in scala a 9 porzioni, risultando non deformati e senza perdere dettagli.

È possibile attivare le guide porzione per la modifica in scala a 9 porzioni nell'ambiente Flash all'interno della finestra di dialogo Converti in simbolo o nella finestra di dialogo Proprietà simbolo. La casella di controllo Attiva guide per modifica in scala a 9 porzioni è disponibile soltanto se si sta pubblicando per Flash Player 8 e il comportamento è impostato su clip filmato. Le guide per la modifica in scala a 9 porzioni non sono disponibili nelle versioni precedenti di Flash o se si sta creando un pulsante o un simbolo grafico. È possibile attivare la modifica in scala a 9 porzioni in ActionScript impostando la proprietà `scale9Grid` su un'istanza di clip filmato.

Se vengono create guide pozione mediante l'interfaccia utente o ActionScript, è possibile tracciare le coordinate `x` e `y`, la larghezza e l'altezza tenendo traccia della proprietà `scale9Grid` del clip filmato.

```
trace(my_mc.scale9Grid); // (x=20, y=20, w=120, h=120)
```

Questo frammento di codice consente di tracciare il valore dell'oggetto Rectangle utilizzato dalla proprietà scale9Grid. Il rettangolo ha coordinate x e y di 20 pixel, una larghezza di 120 pixel e un'altezza di 120 pixel.

## Operazioni con la modifica in scala a 9 porzioni in ActionScript

Nell'esempio seguente, gli strumenti di disegno vengono utilizzati per disegnare un quadrato di 300x300 pixel ridimensionato mediante la modifica in scala a 9 porzioni. Il quadrato viene suddiviso in nove quadrati più piccoli, ognuno dei quali largo 100 pixel e alto 100 pixel. Quando il quadrato viene ridimensionato, i segmenti che non costituiscono angoli vengono estesi in modo da corrispondere alla larghezza e altezza specificate.

### Per modificare in scala a 9 porzioni con ActionScript:

1. Creare un nuovo documento Flash e salvarlo come **ninescale.fla**.
2. Trascinare un componente Button nella libreria del documento corrente.
3. Selezionare lo strumento Rettangolo e disegnare un quadrato rosso (300x300 pixel) con un tratto nero di 15 pixel sullo stage.
4. Selezionare lo strumento Ovale e disegnare un quadrato rosso (50x50 pixel) con un tratto nero di 2 pixel sullo stage.
5. Selezionare il cerchio viola e trascinarlo nell'angolo superiore sinistro del quadrato rosso creato in precedenza.
6. Selezionare lo strumento Ovale e disegnare un nuovo cerchio di circa 200x200 pixel e posizionarlo oltre lo stage.
7. Selezionare il nuovo cerchio sullo stage e trascinarlo in modo che il punto centrale del cerchio sia nell'angolo in basso a sinistra del quadrato.
8. Fare clic al di fuori dell'istanza del cerchio per deselectonare il cerchio.
9. Fare doppio clic sul cerchio per selezionarlo e premere Backspace per eliminare la forma e rimuovere una parte circolare del quadrato.
10. Servendosi del mouse, selezionare l'intero quadrato rosso e il cerchio viola interno.
11. Premere F8 per convertire la forma in un simbolo del clip filmato.
12. Assegnare il nome di istanza **my\_mc** al clip filmato sullo stage.

**13.** Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
import mx.controls.Button;
import flash.geom.Rectangle;

var grid:Rectangle = new Rectangle(100, 100, 100, 100);

var small_button:Button = this.createClassObject(Button, "small_button",
 10, {label:"Small"});
small_button.move(10, 10);
small_button.addEventListener("click", smallHandler);
function smallHandler(eventObj:Object):Void {
 my_mc._width = 100;
 my_mc._height = 100;
}

var large_button:Button = this.createClassObject(Button, "large_button",
 20, {label:"Large"});
large_button.move(120, 10);
large_button.addEventListener("click", largeHandler);
function largeHandler(eventObj:Object):Void {
 my_mc._width = 450;
 my_mc._height = 300;
}

var toggle_button:Button = this.createClassObject(Button,
 "toggle_button", 30, {label:"scale9Grid=OFF", toggle:true,
 selected:false});
toggle_button.move(420, 10);
toggle_button.setSize(120, 22);
toggle_button.addEventListener("click", toggleListener);
function toggleListener(eventObj:Object):Void {
 if (eventObj.target.selected) {
 eventObj.target.label = "scale9Grid=ON";
 my_mc.scale9Grid = grid;
 } else {
 eventObj.target.label = "scale9Grid=OFF";
 my_mc.scale9Grid = undefined;
 }
}
```

Il codice precedente viene suddiviso in cinque sezioni. Nella prima sezione, vengono importate due classi: `mx.controls.Button` (la classe del componente Button) e `flash.geom.Rectangle`. La seconda sezione consente di creare una nuova istanza di classe `Rectangle` e di specificare le coordinate *x* e *y* di 100 pixel così come una larghezza e un'altezza di 100 pixel. Questa istanza del rettangolo viene utilizzata per impostare la griglia della modifica in scala a 9 porzioni di una forma del clip filmato creato in seguito.

Successivamente, viene creata una nuova istanza del componente Button a cui viene assegnato il nome di istanza `small_button`. Ogni qualvolta si fa clic su questo pulsante, il clip filmato creato precedentemente viene ridimensionato a 100 pixel di larghezza e 100 pixel di altezza. La quarta sezione consente di creare una nuova istanza Button denominata `large_button` in modo dinamico . Ogni qualvolta si fa clic su questo pulsante, il clip filmato viene ridimensionato a 450 pixel di larghezza e 300 pixel di altezza. La sezione finale del codice consente di creare una nuova istanza Button che può essere attivata o disattivata. Quando il pulsante è attivato, viene applicata la griglia a 9 porzioni. Se il pulsante è disattivato, la griglia a 9 porzioni risulta disabilitata.

- 14.** Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file SWF.

In questo codice di esempio vengono aggiunte e posizionate tre istanze del componente Button e creati listener di eventi per ogni pulsante. Se si fa clic sul pulsante grande con la griglia a 9 porzioni disattivata, l'immagine viene distorta e appare allungata. Per attivare la griglia a 9 porzioni, fare clic sul pulsante di premuto/non premuto e quindi sul pulsante grande. Attivando la griglia a 9 porzioni, il cerchio nell'angolo superiore sinistro non appare più distorto.



# Creazione di contenuto interattivo con ActionScript

14

Nelle animazioni semplici, Macromedia Flash Player riproduce in sequenza le scene e i fotogrammi di un file SWF. In un file SWF interattivo, usando la tastiera e il mouse gli utenti possono passare alle parti desiderate del file, spostare oggetti, immettere informazioni in moduli ed eseguire molte altre operazioni interattive.

ActionScript viene utilizzato per creare script che comunicano a Flash Player l'azione da eseguire quando si verifica un evento. Alcuni eventi che possono attivare uno script sono quelli che si verificano, ad esempio, quando l'indicatore di riproduzione raggiunge un fotogramma, un clip filmato viene caricato o scaricato oppure l'utente fa clic su un pulsante o preme determinati tasti sulla tastiera.

Gli script possono essere composti da un singolo comando, ad esempio un'istruzione per interrompere la riproduzione di un file SWF, o da una serie di comandi e istruzioni, quali la valutazione di una condizione e la successiva esecuzione di un'azione. Molti comandi ActionScript sono semplici e consentono di creare i comandi di base per un file SWF, mentre alcune azioni richiedono una maggiore familiarità con i linguaggi di programmazione e sono concepite per attività di sviluppo avanzate.

Per ulteriori informazioni sulla creazione di contenuto interattivo con ActionScript, consultare i seguenti argomenti:

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| Informazioni su eventi e interazioni .....                                       | 614 |
| Controllo della riproduzione di file SWF .....                                   | 614 |
| Creazione di interattività ed effetti visivi .....                               | 618 |
| Creazione di associazioni di dati in fase di runtime mediante ActionScript ..... | 632 |
| Analisi di uno script di esempio .....                                           | 641 |

# Informazioni su eventi e interazioni

Un evento viene generato ogni volta che l'utente fa clic con il mouse o preme un tasto della tastiera. Questi tipi di eventi sono generalmente denominati *eventi utente*, poiché vengono generati in risposta a un'azione dell'utente. È possibile utilizzare ActionScript per rispondere a tali eventi, ovvero *gestirli*. Ad esempio, quando un utente fa clic su un pulsante è possibile inviare l'indicatore di riproduzione in un altro fotogramma del file SWF oppure caricare una nuova pagina Web nel browser.

In un file SWF, i pulsanti, i clip filmato e i campi di testo generano eventi ai quali è possibile rispondere. ActionScript fornisce tre modalità di gestione degli eventi: metodi gestore eventi, listener di eventi e gestori `on()` e `onClipEvent()`. Per ulteriori informazioni su eventi e gestione di eventi, consultare il [Capitolo 9, “Gestione degli eventi”](#).

## Controllo della riproduzione di file SWF

Le seguenti funzioni di ActionScript consentono di controllare l'indicatore di riproduzione in una linea temporale e di caricare una nuova pagina Web nella finestra del browser:

- Le funzioni `gotoAndPlay()` e `gotoAndStop()` inviano l'indicatore di riproduzione in un fotogramma o una scena. Si tratta di funzioni globali che è possibile richiamare da qualsiasi script. È inoltre possibile utilizzare i metodi `MovieClip.gotoAndPlay()` e `MovieClip.gotoAndStop()` per navigare nella linea temporale di uno oggetto clip filmato specifico. Vedere [“Passaggio a un fotogramma o a una scena” a pagina 615](#).
- Le azioni `play()` e `stop()` consentono di avviare e interrompere la riproduzione dei i file SWF. Vedere [“Riproduzione e interruzione di clip filmato” a pagina 616](#).
- L'azione `getURL()` consente di passare a un URL differente. Vedere [“Passaggio a un URL diverso” a pagina 617](#).

Per ulteriori informazioni, consultare i seguenti argomenti:

- [“Passaggio a un fotogramma o a una scena” a pagina 615](#)
- [“Riproduzione e interruzione di clip filmato” a pagina 616](#)
- [“Passaggio a un URL diverso” a pagina 617](#)

## Passaggio a un fotogramma o a una scena

Per passare a un determinato fotogramma o a una determinata scena del file SWF, utilizzare le funzioni globali `gotoAndPlay()` o `gotoAndStop()` o i metodi `MovieClip.gotoAndPlay()` e `MovieClip.gotoAndStop()` equivalenti della classe `MovieClip`. Ogni funzione o metodo consente di specificare un fotogramma della scena corrente al quale passare. Se il documento contiene più scene, è possibile specificare la scena e il fotogramma a cui accedere.

L'esempio seguente utilizza la funzione globale `gotoAndPlay()` all'interno di un gestore di eventi `onRelease` di un oggetto pulsante per inviare al fotogramma 10 l'indicatore di riproduzione della linea temporale contenente il pulsante:

```
jump_btn.onRelease = function () {
 gotoAndPlay(10);
};
```

Nell'esempio successivo, il metodo `MovieClip.gotoAndStop()` invia la linea temporale di un'istanza di un clip filmato denominato `categories_mc` al fotogramma 10 e si ferma. Quando si utilizzano i metodi di `MovieClip` `gotoAndPlay()` e `gotoAndStop()`, è necessario specificare un'istanza per il metodo.

```
jump_btn.onPress = function () {
 categories_mc.gotoAndStop(10);
};
```

Nell'esempio finale, la funzione globale `gotoAndStop()` viene utilizzata per spostare l'indicatore di riproduzione dal fotogramma 1 alla scena 2. Se non viene specificata alcuna scena, l'indicatore di riproduzione passa al fotogramma specificato nella scena corrente. È possibile utilizzare il parametro `scene` solo nella linea temporale principale, non all'interno delle linee temporali per clip filmato o altri oggetti del documento.

```
nextScene_mc.onRelease = function() {
 gotoAndStop("Scene 2", 1);
}
```

## Riproduzione e interruzione di clip filmato

Salvo istruzioni diverse, una volta iniziata, la riproduzione del file SWF prosegue per ogni fotogramma della linea temporale. È possibile interrompere o avviare un file SWF utilizzando le funzioni globali `play()` e `stop()` oppure i metodi di MovieClip equivalenti. Ad esempio, è possibile utilizzare la funzione `stop()` per interrompere un file SWF alla fine di una scena prima di procedere a quella successiva. Una volta interrotto, il file SWF deve essere riavviato in modo esplicito, chiamando `play()` o `gotoAndPlay()`.

È possibile usare le funzioni `play()` e `stop()` o i metodi MovieClip per controllare la linea temporale principale o la linea temporale di ciascun clip filmato o file SWF caricato. È necessario che il clip filmato da controllare abbia un nome di istanza e sia presente nella linea temporale.

Il seguente gestore `on(press)` associato a un pulsante avvia l'indicatore di riproduzione nel file SWF o nel clip filmato contenente l'oggetto pulsante:

```
// Associato a un'istanza pulsante
on (press) {
 // Riproduce la linea temporale contenente il pulsante
 play();
}
```

Lo stesso codice del gestore di eventi `on()` produce un risultato diverso se associato a un oggetto clip filmato anziché a un pulsante. Se è associato a un oggetto pulsante, per impostazione predefinita le istruzioni eseguite in un gestore `on()` vengono applicate alla linea temporale contenente il pulsante. Invece, se è associato a un oggetto clip filmato, le istruzioni eseguite in un gestore `on()` sono applicate al clip filmato a cui il gestore `on()` è associato.

Il seguente codice del gestore `onPress()`, ad esempio, ferma la linea temporale del clip filmato a cui il gestore è associato e non quella che contiene il clip filmato:

```
// Associato all'istanza del clip filmato myMovie_mc
myMovie_mc.onPress() {
 stop();
};
```

Le stesse condizioni valgono anche per i gestori `onClipEvent()` associati agli oggetti clip filmato. Il seguente codice, ad esempio, ferma la linea temporale del clip filmato associato al gestore `onClipEvent()` quando il clip viene caricato o viene visualizzato sullo stage per la prima volta:

```
onClipEvent(load) {
 stop();
}
```

## Passaggio a un URL diverso

Per aprire una pagina Web nella finestra di un browser o per trasferire dati a un'altra applicazione presso un URL definito, è possibile utilizzare la funzione globale `getURL()` o il metodo `MovieClip.getURL()`. Ad esempio, è possibile utilizzare un pulsante di collegamento a un nuovo sito Web o inviare variabili della linea temporale a uno script CGI affinché vengano elaborate come se si trattasse di un modulo HTML. È inoltre possibile specificare una finestra di destinazione, analogamente a quanto avviene con una finestra utilizzando il tag di ancoraggio HTML (`<a></a>`).

Il seguente codice, ad esempio, apre la home page macromedia.com in una finestra del browser quando l'utente fa clic sull'istanza del pulsante denominato `homepage_btn`.

```
// Associato al fotogramma
homepage_btn.onRelease = function () {
 getURL("http://www.macromedia.com", "_blank");
};
```

È inoltre possibile inviare variabili insieme all'URL utilizzando i metodi GET o POST. Questa procedura è utile se la pagina che viene caricata da un server applicazioni, ad esempio una pagina di ColdFusion Server (CFM), prevede di ricevere variabili di modulo. Ad esempio, è possibile caricare una pagina CFM denominata `addUser.cfm` che prevede due variabili di modulo, `name` e `age`. Per eseguire questa operazione, è possibile creare un clip filmato denominato `variables_mc` che definisce le due variabili come nell'esempio seguente:

```
variables_mc.firstName = "Francois";
variables_mc.age = 32;
```

Il seguente codice carica quindi `addUser.cfm` in una finestra vuota del browser e passa alla pagina CFM `variables_mc.name` e `variables_mc.age` nell'intestazione POST:

```
variables_mc.getURL("addUser.cfm", "_blank", "POST");
```

La funzionalità `getURL()` dipende dal browser visualizzato. Il modo più affidabile per far sì che i browser funzionino tutti allo stesso modo consiste nel chiamare una funzione JavaScript nel codice HTML che utilizzi il metodo `window.open()` per aprire una finestra. Aggiungere il codice HTML e JavaScript seguente nel modello HTML:

```
<script language="JavaScript">
<!--
 function openNewWindow(myURL) {
 window.open(myURL, "targetWindow");
 }
// -->
</script>
```

È possibile utilizzare il codice ActionScript seguente per chiamare `openNewWindow` dal file SWF:

```
var myURL:String = "http://foo.com";
getURL("javascript:openNewWindow('" + String(myURL) + "');");

```

Per ulteriori informazioni, consultare funzione `getURL` nella *Guida di riferimento di ActionScript*.

## Creazione di interattività ed effetti visivi

Per creare interattività e altri effetti visivi, è necessario comprendere le seguenti tecniche:

- “Creazione di un puntatore personalizzato” a pagina 618
- “Determinazione della posizione del puntatore” a pagina 619
- “Rilevamento di tasti premuti” a pagina 621
- “Impostazione dei valori dei colori” a pagina 624
- “Creazione di controlli audio” a pagina 625
- “Rilevamento di collisioni” a pagina 628
- “Creazione di un semplice strumento di disegno linee” a pagina 630

### Creazione di un puntatore personalizzato

Un puntatore standard è la rappresentazione su schermo della posizione del mouse dell'utente. La sostituzione del puntatore standard del sistema operativo con un puntatore progettato in Flash garantisce una maggiore integrazione dello spostamento del mouse da parte dell'utente con il file SWF. Nell'esempio riportato in questa sezione viene utilizzato un puntatore del mouse personalizzato a forma di grossa freccia. La grande utilità di questa funzione consiste soprattutto nella possibilità di assegnare al puntatore personalizzato qualsiasi forma desiderata, ad esempio un pallone con cui fare goal o un campione di stoffa che può essere appoggiato su un divano per cambiarne il colore.

Per creare un puntatore personalizzato, realizzare il clip filmato del puntatore sullo stage. Successivamente, nascondere in ActionScript il puntatore standard e controllarne il movimento. Per nascondere il puntatore standard, è necessario utilizzare il metodo `hide()` della classe incorporata `Mouse` (vedere `hide` (metodo `Mouse.hide`) nella *Guida di riferimento di ActionScript 2.0*).

### **Per creare un puntatore personalizzato:**

1. Creare un clip filmato da utilizzare come puntatore personalizzato e posizionare un'istanza del filmato sullo stage.
2. Selezionare l'istanza del clip filmato sullo stage.
3. Nella finestra di ispezione Proprietà, digitare **cursor\_mc** nella casella di testo Nome istanza.
4. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, digitare il codice seguente:

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
 this._x = _xmouse;
 this._y = _ymouse;
 updateAfterEvent();
};
```

Il metodo `Mouse.hide()` nasconde il puntatore quando il clip filmato viene visualizzato per la prima volta sullo stage. La funzione `onMouseMove` posiziona il puntatore personalizzato nello stesso punto del puntatore standard e chiama `updateAfterEvent` ogni volta che l'utente sposta il mouse.

La funzione `updateAfterEvent()` aggiorna la schermata immediatamente dopo l'evento specificato e non quando viene disegnato il fotogramma successivo (comportamento predefinito). (Vedere funzione `updateAfterEvent` nella *Guida di riferimento di ActionScript 2.0*.)

5. Selezionare Controllo > Prova filmato per provare il puntatore personalizzato.

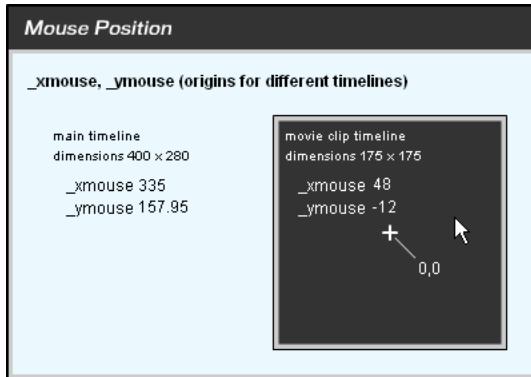
I pulsanti funzionano anche quando si utilizza un puntatore del mouse personalizzato. Si consiglia di inserire il puntatore personalizzato sul primo livello della linea temporale in modo che venga visualizzato in primo piano rispetto ai pulsanti e agli oggetti di altri livelli quando si sposta il mouse nel file SWF. La punta di un puntatore personalizzato rappresenta inoltre il punto di registrazione del clip filmato utilizzato come puntatore personalizzato. Pertanto, se si desidera che una determinata sezione del clip filmato agisca da punta del puntatore, impostare le coordinate del punto di registrazione del clip su tale punto.

Per ulteriori informazioni sui metodi della classe `Mouse`, vedere `Mouse` nella *Guida di riferimento di ActionScript 2.0*.

## Determinazione della posizione del puntatore

Le proprietà `_xmouse` e `_ymouse` possono essere utilizzate per individuare la posizione del puntatore in un file SWF, ad esempio in un'applicazione con mappe che ottiene i valori delle proprietà `_xmouse` e `_ymouse` e li utilizza per calcolare la longitudine e la latitudine di una posizione specifica.

Ogni linea temporale comprende una proprietà `_xmouse` e una proprietà `_ymouse`. Queste proprietà restituiscono la posizione del puntatore all'interno del relativo sistema di coordinate. La posizione è sempre relativa al punto di registrazione. Per la linea temporale principale (`_level0`), il punto di registrazione è l'angolo superiore sinistro. Per un clip filmato, il punto di registrazione dipende da quello impostato alla creazione o sostituzione del clip sullo stage.



*Proprietà `_xmouse` e `_ymouse` all'interno della linea temporale principale e della linea temporale di un clip filmato*

Nella procedura illustrata di seguito vengono indicati modi diversi per ottenere la posizione del puntatore all'interno della linea temporale principale o di un clip filmato.

#### Per ottenere la posizione corrente del puntatore:

1. Creare due campi di testo dinamici e denominarli `box1_txt` e `box2_txt`.
2. Denominare le etichette dei campi di testo, rispettivamente, posizione *x* e posizione *y*.
3. Selezionare Finestra > Azioni per aprire il pannello Azioni, se non è già visualizzato.
4. Aggiungere al file il codice seguente nel riquadro Script:

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
 // Restituisce la posizione X e Y del mouse
 box1_txt.text = _xmouse;
 box2_txt.text = _ymouse;
};
Mouse.addListener(mouseListener);
```

5. Selezionare Controllo > Prova filmato per provare il filmato Flash. I campi `box1_txt` e `box2_txt` indicano la posizione del puntatore durante il suo spostamento sullo stage.

Per ulteriori informazioni sulle proprietà `_xmouse` e `_ymouse`, vedere `_xmouse` (proprietà MovieClip.`_xmouse`) e proprietà `_ymouse` (MovieClip.`_ymouse`) nella *Guida di riferimento di ActionScript 2.0*.

## Rilevamento di tasti premuti

Per intercettare il comportamento incorporato relativo alla pressione di tasti in Flash Player, è possibile usare il gestore globale `on()`, come nell'esempio seguente:

```
/* Quando l'utente preme il tasto Freccia sinistra o destra, viene
modificata la trasparenza del clip filmato a cui è associato il gestore.
*/
on (keyPress "<Left>") {
 this._alpha -= 10;
}
on (keyPress "<Right>") {
 this._alpha += 10;
}
```

Selezionare sempre Controllo > Disattiva scelte rapide da tastiera. In caso contrario, determinati tasti con un comportamento incorporato non vengono ignorati quando si prova l'applicazione con il comando Controllo > Prova filmato. Vedere il parametro `keyPress` di `on` handler nella *Guida di riferimento di ActionScript 2.0*.

È possibile utilizzare i metodi della classe `Key` incorporata per rilevare l'ultimo tasto premuto dall'utente. La classe `Key` non richiede una funzione di costruzione; per utilizzarne i metodi, è sufficiente chiamare i metodi sulla classe, come indicato nel seguente esempio:

```
Key.getCode();
```

È possibile ottenere i codici tasto virtuale oppure i valori ASCII (American Standard Code for Information Interchange) dei tasti premuti:

- Per ottenere il codice tasto virtuale dell'ultimo tasto premuto, utilizzare il metodo `getCode()`.
- Per ottenere il valore ASCII dell'ultimo tasto premuto, utilizzare il metodo `getAscii()`.

Il codice tasto virtuale viene assegnato a ciascun tasto fisico di una tastiera. Il codice tasto virtuale del tasto Freccia sinistra, ad esempio, è 37. Utilizzando un codice tasto virtuale si garantisce che i controlli del file SWF siano gli stessi su tutte le tastiere, indipendentemente dalla lingua o dalla piattaforma.

I valori ASCII sono assegnati ai primi 127 caratteri di ogni set di caratteri e forniscono informazioni sul carattere visualizzato sullo schermo. Ad esempio, la lettera "A" e la lettera "a" hanno valori ASCII diversi.

Per stabilire quali tasti utilizzare e determinare i codici tasto virtuali, adottare uno degli approcci seguenti:

- L'elenco dei codici tasto è riportato nell'[Appendice C, “Tasti della tastiera e valori dei codici tasto”](#).
- Utilizzare una costante della classe Key. Nella casella degli strumenti Azioni, fare clic su Classi ActionScript 2.0 > Filmato > Key > Costanti.
- Assegnare il seguente gestore `onClipEvent()` a un clip filmato, quindi selezionare Controllo > Prova filmato e premere il tasto desiderato:

```
onClipEvent(keyDown) {
 trace(Key.getCode());
}
```

Il codice tasto del tasto desiderato viene visualizzato nel pannello Output.

I metodi della classe Key vengono generalmente usati in combinazione con un gestore di eventi. Nell'esempio seguente, l'utente sposta l'auto utilizzando i tasti freccia. Il metodo `Key.isDown()` indica se il tasto premuto è la freccia destra, sinistra, su o giù. Il gestore di eventi `Key.onKeyDown` determina il valore `Key.isDown(keyCode)` dalle istruzioni `if`. In base al valore, il gestore comunica a Flash Player di aggiornare la posizione dell'auto e di visualizzare la direzione.

L'esempio seguente mostra come rilevare i tasti premuti per spostare un clip filmato in alto, in basso, a sinistra o a destra sullo stage, in base al tasto freccia corrispondente (su, giù, sinistra o destra) che viene premuto. Inoltre un campo di testo visualizza il nome del tasto premuto.

#### **Per creare un clip filmato attivato da tastiera:**

1. Sullo stage, creare un clip filmato che si possa spostare in base ai tasti freccia premuti.  
In questo esempio, il nome dell'istanza del clip filmato è `car_mc`.
2. Selezionare il fotogramma 1 nella linea temporale; selezionare quindi Finestra > Azioni per aprire il pannello Azioni nel caso in cui non sia già visualizzato.
3. Per impostare lo spostamento dell'auto sullo schermo a ogni pressione di un tasto, definire una variabile `distance` e impostarne il valore iniziale su 10:  
`var distance:Number = 10;`
4. Aggiungere il codice seguente al codice esistente nel pannello Azioni:  
`this.createTextField("display_txt", 999, 0, 0, 100, 20);`

5. Per creare il gestore evento per il clip filmato dell'auto che controlla quale tasto freccia (sinistra, destra, su o giù) è correntemente premuto, aggiungere il seguente codice al pannello Azioni:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
};
Key.addListener(keyListener);
```

6. Per verificare se il tasto Freccia destra viene premuto e spostare di conseguenza il clip filmato dell'auto, aggiungere il codice necessario al corpo del gestore di eventi onEnterFrame.

Il codice deve essere simile a quello seguente (il nuovo codice è in grassetto):

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
 if (Key.isDown(Key.LEFT)) {
 car_mc._x = Math.max(car_mc._x - distance, 0);
 display_txt.text = "Left";
 }
};
Key.addListener(keyListener);
```

Se si preme il tasto Freccia sinistra, la proprietà `_x` dell'auto viene impostata sul valore maggiore tra il valore di `_x` corrente meno la distanza e il valore 0. Il valore della proprietà `_x` non può quindi mai essere minore di 0. Nel file SWF viene inoltre visualizzata la parola *Left*.

7. Utilizzare codice analogo per verificare se vengono premuti i tasti Freccia destra, Freccia su o Freccia giù.

Il codice completo sarà simile a quello seguente (il nuovo codice è in grassetto):

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
 if (Key.isDown(Key.LEFT)) {
 car_mc._x = Math.max(car_mc._x - distance, 0);
 display_txt.text = "Left";
 } else if (Key.isDown(Key.RIGHT)) {
 car_mc._x = Math.min(car_mc._x + distance, Stage.width -
car_mc._width);
 display_txt.text = "Right";
 } else if (Key.isDown(Key.UP)) {
 car_mc._y = Math.max(car_mc._y - distance, 0);
 display_txt.text = "Up";
 } else if (Key.isDown(Key.DOWN)) {
```

```

 car_mc._y = Math.min(car_mc._y + distance, Stage.height -
car_mc._height);
 display_txt.text = "Down";
 }
};

Key.addListener(keyListener);

```

8. Selezionare Controllo > Prova filmato per provare il file.

Per ulteriori informazioni sui metodi della classe Key, vedere Key nella *Guida di riferimento di ActionScript 2.0*.

## Impostazione dei valori dei colori

È possibile utilizzare i metodi della classe ColorTransform incorporata (flash.geom.ColorTransform) per modificare il colore di un clip filmato. La proprietà `rgb` della classe ColorTransform assegna al clip filmato i valori esadecimali rosso, verde e blu (RGB). L'esempio seguente usa la proprietà `rgb` per cambiare il colore di un oggetto in base al pulsante scelto dell'utente.

### Per impostare il valore del colore di un clip filmato:

1. Creare un nuovo documento Flash e salvarlo come **setrgb.fla**.
2. Selezionare lo strumento Rettangolo e disegnare un grande quadrato sullo stage.
3. Convertire la forma in un simbolo del clip filmato e assegnare il nome di istanza **car\_mc** nella finestra di ispezione Proprietà.
4. Creare un simbolo di pulsante colorChip, inserire quattro istanze del pulsante sullo stage e denominarle **red\_btn**, **green\_btn**, **blue\_btn** e **black\_btn**.
5. Selezionare il fotogramma 1 nella linea temporale principale e selezionare Finestra > Azioni.
6. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```

import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;

```

7. Per consentire al pulsante blu di modificare il colore del clip filmato **car\_mc** sul blu, aggiungere il seguente codice al pannello Azioni:

```

blue_btn.onRelease = function() {
 colorTrans.rgb = 0x333399; // blu
 trans.colorTransform = colorTrans;
};

```

Il frammento di codice precedente modifica la proprietà `rgb` dell'oggetto di trasformazione del colore e applica di nuovo l'effetto di trasformazione del colore al clip filmato `car_mc` ogni qualvolta viene premuto il pulsante.

8. Ripetere il punto 7 per gli altri pulsanti (`red_btn`, `green_btn` e `black_btn`) per cambiare il colore del clip filmato nel colore corrispondente.

Il codice sarà analogo a quello seguente (il nuovo codice è in grassetto):

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;

blue_btn.onRelease = function() {
 colorTrans.rgb = 0x333399; // blu
 trans.colorTransform = colorTrans;
};
red_btn.onRelease = function() {
 colorTrans.rgb = 0xFF0000; // rosso
 trans.colorTransform = colorTrans;
};
green_btn.onRelease = function() {
 colorTrans.rgb = 0x006600; // verde
 trans.colorTransform = colorTrans;
};
black_btn.onRelease = function() {
 colorTrans.rgb = 0x000000; // nero
 trans.colorTransform = colorTrans;
};
```

9. Selezionare Controllo > Prova filmato per cambiare il colore del clip filmato.

Per informazioni ulteriori sui metodi della classe `ColorTransform`, vedere `ColorTransform` (`flash.geom.ColorTransform`) nella *Guida di riferimento ActionScript 2.0*.

## Creazione di controlli audio

Utilizzare la classe `Sound` incorporata per controllare l'audio in un file SWF. Per utilizzare i metodi della classe `Sound`, è necessario dapprima creare un oggetto `Sound`, quindi è possibile utilizzare il metodo `attachSound()` per inserire in un file SWF un suono della libreria mentre il file è in esecuzione.

Il metodo `setVolume()` della classe `Sound` controlla il volume, mentre il metodo `setPan()` regola il bilanciamento destro e sinistro dell'audio.

Nelle procedure seguenti, viene illustrato come creare controlli audio.

### **Per associare un suono a una linea temporale:**

1. Selezionare File > Importa per importare un suono.
2. Selezionare il suono nella libreria, fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh), quindi scegliere Concatenamento.
3. Selezionare Esporta per ActionScript ed Esporta nel primo fotogramma; quindi assegnare al suono l'identificatore `a_thousand_ways`.
4. Aggiungere un pulsante allo stage e denominarlo `play_btn`.
5. Aggiungere un pulsante allo stage e denominarlo `stop_btn`.
6. Selezionare il fotogramma 1 nella linea temporale principale e selezionare Finestra > Azioni.

Aggiungere il codice seguente nel pannello Azioni:

```
var song_sound:Sound = new Sound();
song_sound.attachSound("a_thousand_ways");
play_btn.onRelease = function() {
 song_sound.start();
};
stop_btn.onRelease = function() {
 song_sound.stop();
};
```

Questo codice innanzitutto interrompe il clip filmato speaker. crea un nuovo oggetto Sound (`song_sound`) e associa il suono il cui identificatore di concatenamento è `a_thousand_ways`. I gestori di eventi `onRelease` associati agli oggetti `playButton` e `stopButton` avviano e interrompono l'audio utilizzando i metodi `Sound.start()` e `Sound.stop()` e inoltre riproducono e interrompono l'audio associato.

7. Selezionare Controllo > Prova filmato per ascoltare l'audio.

### **Per creare un controllo scorrevole del volume:**

1. Utilizzare lo strumento Rettangolo e disegnare un piccolo rettangolo (circa 30 pixel di altezza per 10 pixel di larghezza) sullo stage.
  2. Selezionare lo strumento Selezione e fare doppio clic sulla forma nello stage.
  3. Premere F8 per aprire la finestra di dialogo Converti in simbolo.
  4. Selezionare il tipo Button, immettere il nome di simbolo **volume** e fare clic su OK.
  5. Dopo aver selezionato il simbolo del pulsante sullo stage, immettere il nome di istanza `handle_btn` nella finestra di ispezione Proprietà.
  6. Selezionare il pulsante e scegliere Elabora > Converti in simbolo.
- Selezionare il comportamento del clip filmato. In questo modo, si crea un clip filmato con il pulsante nel fotogramma 1.

7. Selezionare il clip filmato e immettere **volume\_mc** come nome dell'istanza nella finestra di ispezione Proprietà.
8. Selezionare il fotogramma 1 nella linea temporale principale e selezionare Finestra > Azioni.
9. Immettere il codice seguente nel pannello Azioni:

```

this.createTextField("volume_txt", 10, 30, 30, 200, 20);
volume_mc.top = volume_mc._y;
volume_mc.bottom = volume_mc._y;
volume_mc.left = volume_mc._x;
volume_mc.right = volume_mc._x + 100;
volume_mc._x += 100;

volume_mc.handle_btn.onPress = function() {
 startDrag(this._parent, false, this._parent.left, this._parent.top,
 this._parent.right, this._parent.bottom);
};

volume_mc.handle_btn.onRelease = function() {
 stopDrag();
 var level:Number = Math.ceil(this._parent._x - this._parent.left);
 this._parent._parent.song_sound.setVolume(level);
 this._parent._parent.volume_txt.text = level;
};

volume_mc.handle_btn.onReleaseOutside = slider_mc.handle_btn.onRelease;

```

I parametri `startDrag()` `left`, `top`, `right` e `bottom` sono variabili impostate in un'azione del clip filmato.

10. Scegliere Controllo > Prova filmato per utilizzare il cursore del volume.

#### **Per creare un controllo scorrevole del bilanciamento:**

1. Utilizzare lo strumento Rettangolo e disegnare un piccolo rettangolo (circa 30 pixel di altezza per 10 pixel di larghezza) sullo stage.
  2. Selezionare lo strumento Selezione e fare doppio clic sulla forma nello stage.
  3. Premere F8 per aprire la finestra di dialogo Converti in simbolo.
  4. Selezionare il tipo Button, immettere il nome di simbolo **balance** e fare clic su OK.
  5. Dopo aver selezionato il simbolo del pulsante sullo stage, immettere il nome di istanza **handle\_btn** nella finestra di ispezione Proprietà.
  6. Selezionare il pulsante e scegliere Elabora > Converti in simbolo.
- Selezionare il comportamento del clip filmato. In questo modo, si crea un clip filmato con il pulsante nel fotogramma 1.
7. Selezionare il clip filmato e immettere **balance\_mc** come nome dell'istanza nella finestra di ispezione Proprietà.

**8.** Immettere il codice seguente nel pannello Azioni:

```
balance_mc.top = balance_mc._y;
balance_mc.bottom = balance_mc._y;
balance_mc.left = balance_mc._x;
balance_mc.right = balance_mc._x + 100;
balance_mc._x += 50;
balance_mc.handle_btn.onPress = function() {
 startDrag(this._parent, false, this._parent.left, this._parent.top,
 this._parent.right, this._parent.bottom);
};
balance_mc.handle_btn.onRelease = function() {
 stopDrag();
 var level:Number = Math.ceil((this._parent._x - this._parent.left -
 50) * 2);
 this._parent._parent.song_sound.setPan(level);
};
balance_mc.handle_btn.onReleaseOutside =
 balance_mc.handle_btn.onRelease;
```

I parametri `startDrag()` `left`, `top`, `right` e `bottom` sono variabili impostate in un'azione del clip filmato.

**9.** Scegliere Controllo > Prova filmato per utilizzare il cursore del bilanciamento.

Per ulteriori informazioni sui metodi della classe Sound, vedere Sound nella *Guida di riferimento di ActionScript 2.0*.

## Rilevamento di collisioni

Il metodo `hitTest()` della classe MovieClip rileva collisioni in un file SWF. Esso verifica se un oggetto è entrato in collisione con un clip filmato e restituisce un valore booleano (`true` o `false`).

È possibile che sia necessario verificare la presenza di una collisione per accettare se l'utente ha raggiunto una determinata area statica sullo stage o per determinare se un clip filmato ha raggiunto un altro clip. È possibile determinare questi risultati utilizzando `hitTest()`.

È possibile usare i parametri del metodo `hitTest()` per specificare le coordinate `x` e `y` di un'area attiva sullo Stage o usare il percorso `target` di un altro clip filmato come area attiva. Specificando `x` e `y`, `hitTest()` restituisce `true` se il punto identificato da  $(x, y)$  non è trasparente. Quando si passa un `target` al metodo `hitTest()`, i riquadri di limitazione dei due clip filmato vengono confrontati. Se si intersecano, `hitTest()` restituisce `true`. Se i due riquadri non si intersecano, `hitTest()` restituisce `false`.

È inoltre possibile utilizzare `hitTest()` per rilevare una collisione tra due clip filmato.

L'esempio seguente mostra come rilevare una collisione tra un mouse e i clip filmato sullo stage.

**Per rilevare la collisione tra un clip filmato e il puntatore del mouse:**

1. Selezionare il primo fotogramma sul livello 1 della linea temporale.
2. Selezionare Finestra > Azioni per aprire il pannello Azioni, se non è già visualizzato.
3. Immettere il codice seguente nel pannello Azioni:

```
this.createEmptyMovieClip("box_mc", 10);
with (box_mc) {
 beginFill(0xFF0000, 100);
 moveTo(100, 100);
 lineTo(200, 100);
 lineTo(200, 200);
 lineTo(100, 200);
 lineTo(100, 100);
 endFill();
}

this.createTextField("status_txt", 999, 0, 0, 100, 22);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
 status_txt.text = _level0.hitTest(_xmouse, _ymouse, true);
}
Mouse.addListener(mouseListener);
```

4. Scegliere Controllo > Prova filmato e spostare il puntatore del mouse sul clip filmato per provare la collisione.

Il valore `true` viene visualizzato ogni volta che il puntatore si trova su un pixel non trasparente.

**Per rilevare la collisione di due clip filmato:**

1. Trascinare due clip filmato sullo stage e assegnare ad essi i nomi di istanza `car_mc` e `area_mc`.
2. Selezionare il fotogramma 1 della linea temporale.
3. Selezionare Finestra > Azioni per aprire il pannello Azioni, se non è già visualizzato.

- 4.** Immettere il codice seguente nel pannello Azioni:

```
this.createTextField("status_txt", 999, 10, 10, 100, 22);
area_mc.onEnterFrame = function() {
 status_txt.text = this.hitTest(car_mc);
};

car_mc.onPress = function() {
 this.startDrag(false);
 updateAfterEvent();
};
car_mc.onRelease = function() {
 this.stopDrag();
};
```

- 5.** Scegliere Controllo > Prova filmato e trascinare il clip filmato per provare il rilevamento della collisione.

Ogni volta che il riquadro di delimitazione dell'auto interseca il riquadro di delimitazione dell'area, lo stato è true.

Per ulteriori informazioni, consultare hitTest (metodo MovieClip.hitTest) nella *Guida di riferimento di ActionScript 2.0*.

## Creazione di un semplice strumento di disegno linee

Per disegnare linee e riempimenti sullo stage durante l'esecuzione di un file SWF è possibile utilizzare i metodi della classe MovieClip. In questo modo si possono creare strumenti di disegno per gli utenti e disegnare forme nel file SWF in risposta a eventi. I metodi di disegno sono beginFill(), beginGradientFill(), clear(), curveTo(), endFill(), lineTo(), lineStyle() e moveTo().

È possibile applicare questi metodi a qualsiasi istanza di clip filmato, ad esempio myClip.lineTo(), oppure a un livello (\_level10.curveTo()).

I metodi lineTo() e curveTo() consentono di disegnare linee e curve, rispettivamente. Specificare colore, spessore e impostazione alfa per una linea o curva con il metodo lineStyle(). Il metodo di disegno moveTo() imposta la posizione di disegno corrente sulle coordinate x e y dello stage specificate.

I metodi beginFill() e beginGradientFill() riempiono un percorso chiuso rispettivamente con un riempimento uniforme o sfumato e endFill() applica il riempimento specificato nell'ultima chiamata a beginFill() o beginGradientFill(). Il metodo clear() cancella il disegno nell'oggetto clip filmato specificato.

**Per creare un semplice strumento di disegno linee:**

1. In un nuovo documento, creare un pulsante sullo stage e immettere clear\_btn come nome di istanza nella finestra di ispezione Proprietà.
2. Selezionare il fotogramma 1 nella linea temporale.
3. Selezionare Finestra > Azioni per aprire il pannello Azioni, se non è già visualizzato.
4. Nel pannello Azioni, inserire il codice seguente:

```
this.createEmptyMovieClip("canvas_mc", 999);
var isDrawing:Boolean = false;
//
clear_btn.onRelease = function() {
 canvas_mc.clear();
};

var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
 canvas_mc.lineStyle(5, 0xFF0000, 100);
 canvas_mc.moveTo(_xmouse, _ymouse);
 isDrawing = true;
};
mouseListener.onMouseMove = function() {
 if (isDrawing) {
 canvas_mc.lineTo(_xmouse, _ymouse);
 updateAfterEvent();
 }
};
mouseListener.onMouseUp = function() {
 isDrawing = false;
};
Mouse.addListener(mouseListener);
```

5. Selezionare Controllo > Prova filmato per provare il documento.
6. Trascinare il puntatore del mouse per tracciare una linea sullo stage.
7. Fare clic sul pulsante per cancellare il disegno eseguito.

# Creazione di associazioni di dati in fase di runtime mediante ActionScript

Se si creano applicazioni usando i componenti, è spesso necessario aggiungere delle associazioni tra quei componenti in modo da poter interagire con i dati o per far sì che i vari componenti possano interagire tra loro. L'interazione tra componenti è necessaria per creare forme utilizzabili o interfacce con cui gli utenti possono interagire. Per aggiungere associazioni tra i componenti sullo stage, è possibile usare la scheda Associazioni nella finestra di ispezione dei componenti; la stessa può essere usata per associare i dati tra i componenti sullo stage.

Per ulteriori informazioni sull'uso della scheda Associazioni, consultare “[Operazioni con le associazioni nella scheda Associazioni \(solo Flash Professional\)](#)” a pagina 458 nella guida *Uso di Flash*. Per ulteriori informazioni vedere gli articoli in linea seguenti: [Building a Tip of the day Application \(Part 2\)](#), [Data Binding in Macromedia Flash MX Professional 2004](#) e [Building a Google Search Application with Macromedia Flash MX Professional](#).

Le associazioni tra i componenti possono essere create anche usando ActionScript invece della scheda Associazioni; l'aggiunta di codice è spesso un metodo più rapido ed efficace rispetto all'uso dell'ambiente di creazione. La creazione di associazioni mediante ActionScript è necessaria quando si usa il codice per aggiungere componenti a un'applicazione. È possibile scegliere di aggiungere i componenti sullo stage in modo dinamico mediante il metodo `createClassObject()`; tuttavia, in questo caso, non sarebbe possibile creare la scheda Associazioni per creare un'associazione in quanto il componente non esiste fino alla fase di runtime. L'uso di ActionScript per aggiungere associazioni di dati è spesso chiamato *associazione di dati in fase di runtime*.

Per ulteriori informazioni, consultare i seguenti argomenti:

- [Creazione di associazioni tra componenti dell'interfaccia utente mediante ActionScript](#)
- [“Uso dei componenti, delle associazioni e dei formatter personalizzati” a pagina 637](#)
- [“Aggiunta e associazione dei componenti sullo stage” a pagina 640](#)

## Creazione di associazioni tra componenti dell'interfaccia utente mediante ActionScript

L'associazione di dati tra due componenti in fase di runtime è un'operazione piuttosto semplice. Questa operazione è eseguibile in Flash Basic 8 o Flash Professional 8. Per far sì che funzioni, è necessario includere il componente DataBindingClasses nel documento, in quanto contiene le classi necessarie per eseguire le operazioni.

## Per creare un'associazione tra due componenti TextInput mediante ActionScript:

1. Creare un nuovo documento Flash denominato **panel\_as.fla**.
2. Trascinare due copie del componente TextInput sullo stage.
3. Assegnare ai componenti i nomi di istanza seguenti: **in\_ti** e **out\_ti**.
4. Selezionare Finestra > Librerie comuni > Classi e aprire la nuova libreria comune denominata **Classes.fla**.
5. Trascinare una copia del componente DataBindingClasses nel pannello Libreria, oppure trascinare il componente sullo stage e cancellarlo.

Al termine di questa operazione, la libreria comune può essere chiusa. Dopo che è stato cancellato il componente DataBindingClasses dallo stage, Flash ne lascia una copia nella libreria.

### SUGGERIMENTO

Se si dimentica di cancellare il componente DataBindingClasses dallo stage, l'icona del componente rimane visibile in fase di runtime.

### NOTA

Quando si crea un'associazione mediante la finestra di ispezione dei componenti, come nell'esempio precedente, Flash aggiunge il componente DataBindingClasses automaticamente al file FLA. Quando si creano le associazioni di dati mediante ActionScript, è necessario copiare manualmente questa classe nella libreria, come illustrato nel passaggio seguente.

6. Inserire un nuovo livello e denominarlo **actions**.

7. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
var src:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
dest.component = out_ti;
dest.property = "text";
new mx.data.binding.Binding(src, dest);
```

Se si preferisce la versione abbreviata, è possibile importare le classi di associazioni e usare il codice seguente:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
new Binding(src, dest);
```

Questo codice ActionScript crea due punti finali dell'associazione dei dati, uno per ciascun componente di cui si sta eseguendo l'associazione. Il primo punto finale creato indica il componente da cui si sta eseguendo l'associazione (`in_ti`), la proprietà da osservare (`text`) e l'evento che attiverà l'associazione (`focusOut`). Il secondo punto finale creato elenca solo il componente e la proprietà (rispettivamente, `out_ti` e `text`). Infine, si crea l'associazione tra i due punti finali quando si chiama la funzione di costruzione della classe `Binding` (`new Binding(src, dest)`).

Come si vede dal primo frammento di codice, non è necessario usare i nomi completi delle classi (ad esempio `mx.data.binding.EndPoint`) in ActionScript. Se si usa l'istruzione `import` all'inizio del codice è possibile evitare l'uso dei nomi completi. Quando si importano tutte le classi del pacchetto `mx.data.binding` mediante il carattere `*` (il pacchetto include sia la classe `EndPoint` che la classe `Binding`), è possibile abbreviare il codice e fare riferimento direttamente alle classi `EndPoint` e `Binding`. Per ulteriori informazioni sulle istruzioni di importazione, vedere la sezione sull'importazione nella *Guida di riferimento di ActionScript 2.0*.

8. Selezionare Controllo > Prova filmato per provare il codice nell'ambiente di prova. Inserire del testo nel campo di testo `in_ti`.

Quando l'istanza `in_ti` non è più attiva (fare clic sullo stage, premere il tasto Tab, o fare clic sul secondo campo), Flash copia il testo immesso in `in_ti` nel campo di testo `out_ti`.

9. Per salvare le modifiche apportate, selezionare File > Salva.

Se si desidera modificare il testo nel campo di inserimento `out_ti` rispetto all'esercizio precedente, è possibile che il codice diventi un po' più complicato. Se si impostano le associazioni mediante la finestra di ispezione dei componenti, per impostazione predefinita viene creata una connessione bidirezionale. Pertanto, quando si modifica il contenuto di uno dei due campi di testo sullo stage, anche il contenuto dell'altro campo di testo viene automaticamente modificato. Quando si creano le associazioni mediante ActionScript, l'applicazione funziona nel modo opposto. Le associazioni di dati in fase di runtime, per impostazione predefinita, sono unidirezionali, come illustrato nell'esempio seguente.

Per creare un'associazione bidirezionale mediante ActionScript, è necessario apportare un paio di piccole modifiche ai frammenti di codice indicati nella procedura precedente. Questo esempio usa il secondo frammento di codice ActionScript abbreviato del punto 7.

#### Per creare un'associazione bidirezionale:

1. Aprire il file **panel\_as.fla** dell'esempio precedente.
2. Modificare leggermente il codice ActionScript (vedere il codice in **grassetto**) in modo che corrisponda a quanto segue:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
dest.event = "focusOut";
new Binding(src, dest, null, true);
```

Le due modifiche apportate al codice ActionScript sortiscono gli effetti seguenti:

- Definiscono una proprietà evento per l'istanza EndPoint di destinazione.
- Definiscono due parametri ulteriori per la funzione di costruzione Binding.

Il primo parametro viene usato per le opzioni di formattazione avanzate: il valore può essere impostato su `null` o `undefined`. Il secondo parametro definisce se l'associazione è bidirezionale (`true`) o unidirezionale (`false`).

A questo punto ci si potrebbe chiedere da dove viene l'evento `focusOut` ed è proprio qui che il codice ActionScript si fa più complesso. È possibile cercare nella classe `TextInput` e usare alcuni dei metodi elencati, ad esempio `change()` o `enter()`, ma non sarà possibile trovarvi l'evento `focusOut`. La classe `TextInput` eredita dalle classi `UIObject` e `UIComponent`. Se si visualizza la classe `UIComponent`, che aggiunge il supporto dell'attivazione ai componenti, sono visibili altri quattro eventi: `focusIn`, `focusOut`, `keyDown` e `keyUp`. Questi eventi possono essere usati con il componente `TextInput`.

3. (Opzionale) Se si vuole fare in modo che l'esempio precedente aggiorni il valore nel campo di immissione testo `out_ti`, è possibile cambiare l'evento da `focusOut` a `change`.
4. Selezionare Controllo > Prova filmato per provare il documento.

Flash modifica il secondo valore nel campo di immissione testo `in_ti` e aggiorna il valore relativo di `out_ti`. In questo modo è stata creata correttamente un'associazione bidirezionale.

È possibile utilizzare le classi Binding con la maggior parte dei componenti dell'interfaccia utente della versione 2 di Macromedia Component Architecture, non soltanto col componente TextInput. L'esempio seguente mostra come associare istanze CheckBox e componenti Label in fase di runtime mediante ActionScript.

**Per usare le classi di associazione con il componente CheckBox:**

1. Creare un nuovo documento Flash.
2. Selezionare File > Salva con nome e assegnare al nuovo file il nome **checkbox\_as.fla**.
3. Selezionare Finestra > Librerie comuni > Classi.
4. Trascinare una copia della classe DataBindingClasses nella libreria del documento.
5. Trascinare una copia del componente CheckBox nello stage e assegnarvi il nome di istanza **my\_ch**.
6. Trascinare una copia del componente Label nello stage e assegnarvi il nome di istanza **my\_lbl**.
7. Creare un nuovo livello e denominarlo **actions**.
8. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
var srcEndPoint:Object = {component:my_ch, property:"selected",
 event:"click"};
var destEndPoint:Object = {component:my_lbl, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Per definire i punti finali si usano gli oggetti invece di creare nuove istanze della classe EndPoint, come illustrato negli esercizi precedenti in questa sezione. Il frammento di codice di questo passaggio consente di creare due oggetti, che fungono da punti finali dell'associazione. L'associazione viene creata nel momento in cui si chiama la funzione di costruzione della classe Binding; se si desidera ridurre ulteriormente la quantità di codice (e la leggibilità), è possibile definire gli oggetti in linea come illustrato nel seguente frammento di codice:

```
new mx.data.binding.Binding({component:my_ch, property:"selected",
 event:"click"}, {component:my_lbl, property:"text"});
```

Questo codice ActionScript riduce la leggibilità ma anche la quantità di dati da digitare. Se si condividono i file FLA o ActionScript, si consiglia di usare il primo frammento di codice ActionScript, in quanto è più semplice da leggere.

# Uso dei componenti, delle associazioni e dei formatter personalizzati

I formatter personalizzati consentono di formattare i dati complessi in un determinato modo; inoltre, la formattazione personalizzata consente di visualizzare in modo più semplice le immagini, il testo formattato HTML e altri componenti all'interno di un componente quale DataGrid. L'esempio seguente mostra l'utilità dei formatter personalizzati.

## Per usare i formatter personalizzati in un documento:

1. Creare un nuovo file FLA e aggiungere la classe DataBindingClasses alla libreria (Finestra > Librerie comuni > Classi).
2. Trascinare una copia del componente DateChooser nello stage e assegnarvi il nome di istanza **my\_dc**.
3. Trascinare una copia del componente Label nello stage e assegnarvi il nome di istanza **my\_lbl**.
4. Inserire un nuovo livello e denominarlo **actions**.
5. Aggiungere il seguente codice ActionScript al fotogramma 1 del livello actions:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

Questo codice crea un'associazione tra la proprietà `selectedDate` del componente DateChooser e la proprietà `text` del componente Label sullo stage. Ogni volta che si fa clic su una nuova data del calendario, la data selezionata viene visualizzata nel componente Label.

6. Salvare il documento Flash come **customformat.fla** in una posizione adeguata del disco rigido.

(Verrà riutilizzato nell'esercizio seguente.)

7. Selezionare Controllo > Prova filmato per provare il documento.

Se si cerca di cambiare le date nel componente Calendar, la data attualmente selezionata viene visualizzata nel componente Label, che però non è sufficientemente ampio da visualizzare la data per intero, pertanto il testo viene troncato.

**8.** Chiudere il file SWF di prova per tornare all'ambiente di creazione.

Modificare le dimensioni del componente Label sullo stage o selezionare il componente Label e impostare la proprietà `autoSize` su `left` nella scheda Parametri della finestra di ispezione delle proprietà.

**9.** Selezionare Controllo > Prova filmato per provare nuovamente il documento.

A questo punto il campo di testo visualizza la data per intero, che però ha un aspetto strano e privo di formattazione. A seconda del fuso orario in cui ci si trova e della data selezionata, questa potrebbe essere visualizzata in modo simile al seguente: Thu Nov 4 00:00:00 GMT-0800 2004

Anche se il funzionamento dell'associazione è corretto e viene visualizzata la proprietà `selectedDate`, queste date non sono di facile comprensione e utilizzo da parte degli utenti. Non tutti desiderano visualizzare gli offset del fuso orario ed è possibile che non si desideri nemmeno visualizzare ore, minuti e secondi; è pertanto necessario un modo che consenta di formattare la data in modo che risulti più leggibile e meno "meccanica". I formatter personalizzati sono particolarmente utili per la formattazione del testo.

## Formattazione dei dati con la classe CustomFormatter

La classe CustomFormatter definisce due metodi, `format()` e `unformat()`, che consentono di trasformare i valori dei dati da un tipo di dati specifico a stringa e viceversa. Per impostazione predefinita questi metodi non eseguono alcuna azione; è necessario implementarli in una sottoclasse di `mx.data.binding.CustomFormatter`. La classe CustomFormatter consente di convertire i tipi di dati in stringa e viceversa. In questo caso, si vuole convertire la proprietà `selectedDate` dal componente DateChooser in una stringa formattata correttamente quando il valore viene copiato nel componente Label.

L'esempio seguente mostra come creare un formatter personalizzato che visualizzi la data come NOV 4, 2004 invece di visualizzare la stringa data predefinita.

**NOTA**

È necessario completare l'esercizio fornito nella sezione ["Uso dei componenti, delle associazioni e dei formatter personalizzati"](#) a pagina 637 prima di iniziare questo.

### **Per formattare i dati con la classe CustomFormatter:**

1. Selezionare File > Nuovo e quindi File ActionScript per creare un nuovo file AS.
2. Selezionare File > Salva con nome e salvare il nuovo file come **DateFormat.as**.
3. Immettere il codice seguente nella finestra Script:

```
class DateFormat extends mx.data.binding.CustomFormatter {
 function format(rawValue:Date):String {
 var returnValue:String;
 var monthName_array:Array =
 ["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC"];
 returnValue = monthName_array[rawValue.getMonth()]+"
 "+rawValue.getDate()+", "+rawValue.getFullYear();
 return returnValue;
 }
}
```

La prima sezione del codice definisce la nuova classe denominata **DateFormat**, che estende la classe **CustomFormatter** nel pacchetto **mx.data.binding**. Si ricorda che Flash compila le classi di associazioni nel file del componente **DataBindingClasses**, pertanto non è possibile visualizzarle direttamente o ritrovarle nella cartella **Classes** della directory di installazione di Flash.

L'unico metodo usato è il metodo **format()**, che converte l'istanza della data in un formato di stringa personalizzato. Il passo successivo consiste nella creazione di un array di nomi di mesi, in modo che il risultato finale assomigli a NOV 4, 2004 invece che al formato predefinito della data. Si ricorda che in Flash gli array sono su base zero, pertanto se il valore di **rawValue.getMonth()** restituisce 1, rappresenta febbraio e non gennaio (in quanto gennaio è il mese 0). Il codice rimanente costruisce la stringa di formato personalizzato mediante la concatenazione dei valori e la restituzione della stringa **returnValue**.

È possibile che si verifichino dei problemi quando si eseguono operazioni con le classi incluse in un clip compilato (visualizzabile nel frammento di codice precedente). Poiché si estende una classe situata nella classe **DataBindingClasses** e non disponibile in Flash, quando si verifica la sintassi della classe precedente si rileva il seguente errore:

```
Error <path to DateFormat class>\DateFormat.as: Line 1: The class
'mx.data.binding.CustomFormatter' could not be loaded.
class DateFormat extends mx.data.binding.CustomFormatter {
```

Total ActionScript Errors: 1 Reported Errors: 1

Il codice, molto probabilmente, è corretto; questo problema si verifica nei casi in cui Flash non è in grado di individuare la classe e, per questo motivo, il controllo della sintassi dà risultati negativi.

- 4.** Salvare il file DateFormat.as.
- 5.** Aprire il file customformat.fla creato nell'esercizio della sezione “[Uso dei componenti, delle associazioni e dei formatter personalizzati](#)”. Accertarsi di salvare o copiare DateFormat.as nella stessa directory di questo file.
- 6.** Nel file customformat.fla, modificare il codice ActionScript nel fotogramma 1 del livello actions in modo che corrisponda al seguente:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest, {cls:mx.data.formatters.Custom,
settings:{classname:"DateFormat", classname_class:DateFormat}});
```

In questo caso, si definisce un oggetto customFormatter, il che indica a Flash che si sta usando la classe DateFormat appena creata per formattare il punto finale dell'associazione.

- 7.** Salvare le modifiche apportate al documento e selezionare Controllo > Prova filmato per provare il codice.

## Aggiunta e associazione dei componenti sullo stage

Uno dei grossi vantaggi dell'uso delle classi di associazione con ActionScript consiste nella possibilità di creare associazioni tra i componenti che Flash ha aggiunto allo stage in fase di runtime. Si supponga di creare una classe personalizzata che aggiunge i campi di testo appropriati allo stage in fase di runtime, quindi convalida i dati necessari e aggiunge le associazioni richieste; i componenti presenti nella libreria possono essere aggiunti in modo dinamico ed è possibile creare le associazioni usando poche righe di codice in più.

### Per aggiungere e associare i componenti sullo stage mediante ActionScript:

- 1.** Creare un nuovo documento Flash.
- 2.** Trascinare un componente ComboBox e un componente Label nella libreria del documento.
- 3.** Inserire un nuovo livello e denominarlo **actions**.

**4. Aggiungere il seguente codice al fotogramma 1 del livello actions:**

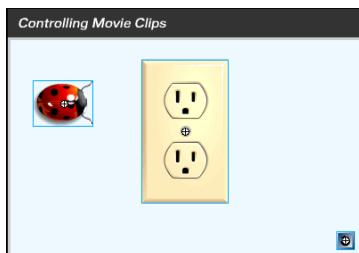
```
import mx.data.binding.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 1, {_x:10,
_y:10});
this.createClassObject(mx.controls.Label, "my_lbl", 2, {_x:10, _y:40});
my_cb.addItem("JAN", 0);
my_cb.addItem("FEB", 1);
my_cb.addItem("MAR", 2);
my_cb.addItem("APR", 3);
my_cb.addItem("MAY", 4);
my_cb.addItem("JUN", 5);
var src:EndPoint = new EndPoint();
src.component = my_cb;
src.property = "value";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

La prima riga del codice ActionScript importa le classi dal pacchetto `mx.data.binding` in modo che non sia necessario usare i percorsi completi nel codice. Le due righe di codice successive associano i componenti della libreria del documento allo stage. Quindi, si posizionano i componenti sullo stage.

Infine, si aggiungono i dati all'istanza ComboBox e si crea l'associazione tra il componente ComboBox `my_cb` e il componente Label `my_lbl` sullo stage.

## Analisi di uno script di esempio

Nel file SWF di esempio zapper.swf, visualizzabile nella guida Uso di Flash, se si trascina l'insetto sulla presa di corrente, l'insetto cade e la presa trema. La linea temporale principale presenta un solo fotogramma e contiene tre oggetti: la coccinella, la presa di corrente e il pulsante di ripristino. Ciascuno di questi oggetti è l'istanza di un clip filmato.



Lo script seguente è associato al fotogramma 1 della linea temporale principale:

```
var initx:Number = bug_mc._x;
var inity:Number = bug_mc._y;
var zapped:Boolean = false;

reset_btn.onRelease = function() {
 zapped = false;
 bug_mc._x = initx;
 bug_mc._y = inity;
 bug_mc._alpha = 100;
 bug_mc._rotation = 0;
};

bug_mc.onPress = function() {
 this.startDrag();
};
bug_mc.onRelease = function() {
 this.stopDrag();
};
bug_mc.onEnterFrame = function() {
 if (this.hitTest(this._parent.zapper_mc)) {
 this.stopDrag();
 zapped = true;
 bug_mc._alpha = 75;
 bug_mc._rotation = 20;
 this._parent.zapper_mc.play();
 }
 if (zapped) {
 bug_mc._y += 25;
 }
};
```

Il nome di istanza della coccinella è `bug_mc`, mentre il nome di istanza della presa di corrente è `zapper_mc`. Nello script si fa riferimento all'insetto con `this` in quanto lo script è associato all'insetto e la parola riservata `this` si riferisce all'oggetto che la contiene.

Sono presenti gestori di eventi per eventi diversi: `onRelease()`, `onPress()` e `onEnterFrame()`. I gestori di eventi vengono definiti nel fotogramma 1 dopo il caricamento del file SWF. Le azioni nel gestore di eventi `onEnterFrame()` vengono eseguite ogni volta che l'indicatore di riproduzione entra in un fotogramma. Anche nei file SWF composti da un solo fotogramma, l'indicatore di riproduzione entra nel fotogramma più volte e lo script viene eseguito ripetutamente.

Vengono definite due variabili, `initx` e `inity`, per memorizzare le posizioni iniziali `x` e `y` dell'istanza del clip filmato `bug_mc`. Viene definita e assegnata una funzione al gestore di eventi `onRelease` dell'istanza `reset_btn`. Questa funzione viene chiamata ogni volta che si preme e si rilascia il pulsante del mouse sul pulsante `reset_btn`. La funzione ricolloca la coccinella nella posizione iniziale sullo stage, ripristina il valore di rotazione e il valore alfa e reimposta la variabile `zapped` su `false`.

Un'istruzione condizionale `if` utilizza il metodo `hitTest()` per verificare se l'istanza della coccinella tocca l'istanza della presa di corrente (`this._parent.zapper_mc`). La valutazione può restituire due risultati: `true` o `false`:

- Se il metodo `hitTest()` restituisce `true`, viene chiamato il metodo `stopDrag()`, la variabile `zapper_mc` viene impostata su `true`, vengono modificate le proprietà `alpha` e `rotation` e viene avviata la riproduzione dell'istanza `zapped`.
- Se il metodo `hitTest()` restituisce `false`, il codice all'interno delle parentesi graffe (`{ }` ) che segue immediatamente l'istruzione `if` non viene eseguito.

Le azioni contenute nell'istruzione `onPress()` vengono eseguite quando si fa clic con il pulsante del mouse sull'istanza `bug_mc`. Le azioni contenute nell'istruzione `onRelease()` vengono eseguite quando si rilascia il pulsante del mouse sull'istanza `bug_mc`.

L'azione `startDrag()` consente di trascinare la coccinella. Poiché lo script è associato all'istanza `bug_mc`, la parola chiave `this` indica che l'istanza trascinabile è `bug`:

```
bug_mc.onPress = function() {
 this.startDrag();
};
```

L'azione `stopDrag()` interrompe l'azione di trascinamento:

```
bug_mc.onRelease = function() {
 this.stopDrag();
};
```



# Operazioni con immagini, audio e video

15

Se durante la creazione di un documento in Macromedia Flash Basic 8 o in Macromedia Flash Professional 8 si importano immagini o suoni, questi vengono compressi e memorizzati nel file SWF al momento della pubblicazione del file. Oltre a importare contenuti multimediali durante la creazione, è possibile caricarli da fonti esterne, compresi altri file SWF, in fase di runtime. La scelta di mantenere i contenuti multimediali al di fuori di un documento Flash può dipendere da diversi motivi.

**Dimensioni ridotte dei file** Se i file multimediali di grandi dimensioni vengono mantenuti al di fuori del documento Flash e caricati in fase di runtime, è possibile ridurre il tempo di scaricamento iniziale per le applicazioni e le presentazioni, in particolare con connessioni Internet lente.

**Creazione di un sistema modulare per presentazioni di grandi dimensioni** È possibile suddividere una presentazione o un'applicazione di grandi dimensioni in più file SWF e caricarli in base alle necessità in fase di runtime. In tal modo non solo si riduce il tempo iniziale di scaricamento, ma si semplificano anche le operazioni di manutenzione e di aggiornamento della presentazione.

**Separazione del contenuto dalla presentazione** Questo approccio è molto diffuso nello sviluppo di applicazioni, in particolare quelle basate su dati. Ad esempio, nel caso di un'applicazione "carrello della spesa", può essere visualizzata un'immagine di ciascun prodotto. Se si caricano le immagini in fase di runtime, è possibile aggiornare semplicemente l'immagine di un prodotto senza modificare il file FLA originale.

**Uso di funzionalità della fase di runtime** Alcune funzionalità, quali la riproduzione di file FLV (Flash Video) e MP3 caricati in modo dinamico, sono disponibili esclusivamente in fase di runtime tramite ActionScript.

Questa sezione descrive come lavorare con i file di immagine, i file audio e i video FLV nelle applicazioni Flash. Per ulteriori informazioni, consultare i seguenti argomenti:

|                                                                                       |     |
|---------------------------------------------------------------------------------------|-----|
| Informazioni su caricamento e lavoro con contenuti multimediali esterni .....         | 646 |
| Caricamento di file SWF e file di immagine esterni .....                              | 647 |
| Informazioni sul caricamento e l'uso di file MP3 esterni .....                        | 652 |
| Assegnazione di identificatori di concatenamento alle risorse della libreria .....    | 658 |
| Informazioni sull'uso dei file video FLV .....                                        | 659 |
| Informazioni sulla creazione di animazioni di avanzamento per file multimediali ..... | 681 |

## Informazioni su caricamento e lavoro con contenuti multimediali esterni

In fase di runtime è possibile caricare in un'applicazione Flash diversi tipi di file multimediali: file SWF, MP3, JPEG, GIF, PNG e FLV. Tuttavia, non tutte le versioni di Flash Player supportano tutti i tipi di file multimediali. Per ulteriori informazioni sui tipi di file di immagine supportati in Macromedia Flash Player 8, vedere “[Caricamento di file SWF e file di immagine esterni](#)” a pagina 647. Per informazioni sul supporto di video FLV in Flash Player, vedere “[Informazioni sull'uso dei file video FLV](#)” a pagina 659.

Macromedia Flash Player consente di caricare contenuti multimediali esterni da qualsiasi indirizzo HTTP o FTP, da un disco locale tramite un percorso relativo o utilizzando il protocollo `file://`.

Per caricare file SWF e file di immagine esterni, è possibile utilizzare la funzione `loadMovie()` o `loadMovieNum()`, il metodo `MovieClip.loadMovie()` o il metodo `MovieClipLoader.loadClip()`. I metodi di classe in genere garantiscono maggiori funzioni e una flessibilità superiore rispetto alle funzioni globali e sono adatti per applicazioni complesse. Se si carica un file SWF o un file di immagine, è necessario specificare un clip filmato o un livello del file SWF come destinazione per tali contenuti multimediali. Per ulteriori informazioni sul caricamento dei file SWF e dei file di immagine, vedere “[Caricamento di file SWF e file di immagine esterni](#)” a pagina 647.

Per riprodurre un file MP3 esterno, utilizzare il metodo `loadSound()` della classe Sound. Questo metodo consente di specificare se il file MP3 può essere scaricato in modo progressivo o deve essere scaricato completamente affinché abbia inizio la riproduzione. È inoltre possibile vedere le informazioni ID3 incluse nei file MP3, se disponibili. Per ulteriori informazioni, vedere “[Lettura dei tag ID3 nei file MP3](#)” a pagina 656.

Flash Video (FLV) è il formato video nativo utilizzato da Flash Player. È possibile riprodurre i file FLV tramite HTTP o dal file system locale. La riproduzione di file FLV esterni offre diversi vantaggi per l'incorporamento di video in un documento Flash, ad esempio migliori prestazioni e gestione della memoria, oltre a frequenze di fotogrammi video e Flash indipendenti. Per ulteriori informazioni, vedere [“Riproduzione dinamica di file FLV esterni” a pagina 662](#).

È inoltre possibile precaricare i contenuti multimediali esterni o tenere traccia dell'avanzamento dello scaricamento. In Flash Player 7 è presente la classe MovieClipLoader, utilizzabile per tenere traccia dell'avanzamento dello scaricamento dei file SWF o dei file di immagine. Per precaricare i file MP3 e FLV, è possibile utilizzare il metodo `getBytesLoaded()` della classe Sound e la proprietà `bytesLoaded` della classe NetStream. Per ulteriori informazioni, vedere [“Precarica di file FLV” a pagina 665](#).

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file immagine in un file SWF. È possibile trovare i file sorgente di esempio, `gallery_tree.fla` e `gallery Tween.fla`, nella cartella Samples sul disco rigido.

- In Windows, accedere a `unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries`.
- In Macintosh, accedere a `Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Galleries`.

## Caricamento di file SWF e file di immagine esterni

Per caricare un file SWF o un file di immagine, utilizzare la funzione globale `loadMovie()` o `loadMovieNum()`, il metodo `loadMovie()` della classe MovieClip o il metodo `loadClip()` della classe MovieClipLoader. Per ulteriori informazioni sul metodo `loadClip()`, vedere `MovieClipLoader.loadClip()` nella *Guida di riferimento di ActionScript 2.0*.

Per quanto riguarda i file di immagine, Flash Player 8 supporta i file di tipo JPEG (progressivi e non), le immagini GIF (trasparenti e non, anche se viene caricato solo il primo fotogramma di un file GIF animato) e i file PNG (trasparenti e non).

Per caricare un file SWF o un file di immagine in un livello di Flash Player, usare la funzione `loadMovieNum()`. Per caricare un file SWF o un file di immagine nella destinazione di un clip filmato, usare la funzione o il metodo `loadMovie()`. In entrambi i casi, il contenuto caricato sostituisce il contenuto del livello o della destinazione del clip filmato specificati.

Se si carica un file SWF o un file di immagine nella destinazione di un clip filmato, l'angolo superiore sinistro del file SWF o dell'immagine viene posizionato sul punto di registrazione del clip filmato. Poiché il punto di registrazione è spesso il centro del clip filmato, il contenuto caricato può non apparire centrato. Inoltre, quando si carica un file SWF o un'immagine su una linea temporale principale, l'angolo superiore sinistro dell'immagine si posiziona sull'angolo superiore sinistro dello stage. Il contenuto caricato eredita dal clip filmato le proprietà di rotazione e scala ma il contenuto originale del clip filmato viene rimosso.

È possibile inviare le variabili di ActionScript tramite una chiamata `loadMovie()` o `loadMovieNum()`. Tale operazione risulta utile se, ad esempio, l'URL specificato nella chiamata del metodo corrisponde a uno script del server che restituisce un file SWF o un file di immagine in base ai dati provenienti dall'applicazione Flash.

Quando si utilizza la funzione globale `loadMovie()` o `loadMovieNum()`, è necessario specificare come parametro il livello o il filmato di destinazione. L'esempio seguente carica il file `contents.swf` dell'applicazione Flash nell'istanza del clip filmato denominata `image_mc`:

```
image_mc.loadMovie("contents.swf");
```

È possibile usare `MovieClip.loadMovie()` per ottenere lo stesso risultato:

```
image_mc.loadMovie("contents.swf");
```

L'esempio seguente carica l'immagine JPEG `image1.jpg` nell'istanza del clip filmato `image_mc`:

```
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Per ulteriori informazioni sul caricamento dei file SWF e dei file di immagine esterni, vedere ["Informazioni sul caricamento di file SWF e sulla linea temporale principale"](#) a pagina 651.

Per precaricare i file SWF e JPEG nelle istanze di clip filmato è possibile usare la classe `MovieClipLoader`. Questa classe è dotata di un listener di eventi che consente di fornire notifiche sullo stato dello scaricamento dei file nei clip filmato. Per utilizzare l'oggetto `MovieClipLoader` per precaricare i file SWF e JPEG è necessario eseguire quanto segue:

**Creazione di un nuovo oggetto `MovieClipLoader`** È possibile utilizzare un unico oggetto `MovieClipLoader` per controllare lo stato di scaricamento di più file oppure creare un oggetto separato per ciascun file. Creare un nuovo clip filmato, caricare i contenuti nel clip, quindi creare l'oggetto `MovieClipLoader` come mostrato nel codice seguente:

```
this.createEmptyMovieClip("img_mc", 999);
var my_mc1:MovieClipLoader = new MovieClipLoader();
```

**Creazione di un oggetto `listener` e di gestori di eventi** L'oggetto `listener` può essere un oggetto ActionScript qualsiasi, ad esempio un oggetto `Object` generico, un clip filmato o un componente personalizzato.

Il seguente esempio crea un oggetto listener generico denominato `loadListener` e definisce per sé le funzioni `onLoadError`, `onLoadStart`, `onLoadProgress` e `onLoadComplete`.

```
// Crea un oggetto listener:
var mc1Listener:Object = new Object();
mc1Listener.onLoadError = function(target_mc:MovieClip, errorCode:String,
status:Number) {
 trace("Error loading image: " + errorCode + " [" + status + "]");
};
mc1Listener.onLoadStart = function(target_mc:MovieClip):Void {
 trace("onLoadStart: " + target_mc);
};
mc1Listener.onLoadProgress = function(target_mc:MovieClip,
numBytesLoaded:Number, numBytesTotal:Number):Void {
 var numPercentLoaded:Number = numBytesLoaded / numBytesTotal * 100;
 trace("onLoadProgress: " + target_mc + " is " + numPercentLoaded + "%
loaded");
};
mc1Listener.onLoadComplete = function(target_mc:MovieClip,
status:Number):Void {
 trace("onLoadComplete: " + target_mc);
};
```

**NOTA**

Flash Player 8 consente di controllare lo stato HTTP di scaricamento di MovieClipLoader all'interno dei listener di eventi `onLoadComplete` e `onLoadError`. Questa funzione permette di verificare perché non è stato possibile scaricare il file: se si è verificato un errore del server, se non è stato possibile trovare il file e così via.

**Registrazione dell'oggetto listener con l'oggetto MovieClipLoader** Perché l'oggetto `listener` sia in grado di ricevere gli eventi di caricamento, registrarlo con l'oggetto `MovieClipLoader` come nel codice seguente:

```
my_mcl.addListener(mc1Listener);
```

**Avvio del caricamento del file (di immagine o SWF) nel clip target** Per avviare lo scaricamento del file di immagine o SWF, usare il metodo `MovieClipLoader.loadClip()` come illustrato nel codice seguente:

```
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
img_mc);
```

**NOTA**

Per controllare lo stato dello scaricamento dei file caricati con il metodo `MovieClipLoader.loadClip()`, è possibile utilizzare esclusivamente i metodi `MovieClipLoader`. Non è possibile utilizzare invece la funzione `loadMovie()` o il metodo `MovieClip.loadMovie()`.

Nell'esempio seguente, il metodo `setProgress()` del componente `ProgressBar` viene utilizzato per visualizzare lo stato di scaricamento di un file SWF.  
(Vedere “`ProgressBar.setProgress()`” nella *Guida di riferimento dei componenti*.)

## **Procedura per la visualizzazione dello stato di scaricamento mediante il componente ProgressBar:**

1. Creare un nuovo documento Flash e salvarlo come **progress.fla**.
2. Aprire il pannello Componenti (Finestra > Componenti).
3. Trascinare il componente ProgressBar dal pannello Componenti nello stage.
4. Nella finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà), assegnare al componente ProgressBar il nome **my\_pb**.
5. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
6. Aggiungere il codice seguente nel pannello Azioni:

```
var my_pb:mx.controls.ProgressBar;
my_pb.mode = "manual";

this.createEmptyMovieClip("img_mc", 999);

var my_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
 my_pb.label = "loading: " + target_mc._name;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
 numBytesLoaded:Number, numBytesTotal:Number):Void {
 var pctLoaded:Number = Math.ceil(100 * (numBytesLoaded /
 numBytesTotal));
 my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 img_mc);
```

7. Provare il documento selezionando Controllo > Prova filmato.

L'immagine viene caricata nel clip filmato **img\_mc**.

8. Selezionare File > Impostazioni pubblicazione > Formati e assicurarsi che le opzioni SWF e HTML siano selezionate.
9. Fare clic su Pubblica e trovare i file HTML e SWF sul disco rigido.

Si trovano nella stessa cartella del file **progress.fla** salvato nel passaggio 1.

- 10.** Fare doppio clic sul documento HTML per aprirlo in un browser e vedere l'animazione della barra di avanzamento.

**NOTA**

Per visualizzare correttamente la barra di avanzamento quando si caricano file nell'ambiente di prova, caricare sempre i file da Internet e non dalla cache locale. Il caricamento di un file locale è troppo rapido e non consente di visualizzare l'avanzamento. In alternativa, caricare il file SWF e provare il documento su un server.

Per ulteriori informazioni, vedere “[Informazioni sul caricamento di file SWF e sulla linea temporale principale](#)” a pagina 651. Per ulteriori informazioni sulla classe MovieClipLoader, vedere MovieClipLoader nella *Guida di riferimento di ActionScript 2.0*. Per informazioni sulla creazione di un'animazione per la barra di avanzamento, vedere “[Creazione di un'animazione di avanzamento per caricare file SWF e di immagine](#)” a pagina 682.

È possibile trovare esempi di applicazioni di gallerie fotografiche sul disco rigido. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file di immagine in un file SWF. È possibile trovare i file sorgente di esempio, gallery\_tree.fla e gallery\_tween.fla, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Galleries*.

## Informazioni sul caricamento di file SWF e sulla linea temporale principale

La proprietà ActionScript `_root` specifica o restituisce un riferimento alla linea temporale principale di un file SWF. Se si carica un file SWF in un clip filmato che si trova in un altro file SWF, tutti i riferimenti a `_root` presenti nel file SWF caricato vengono risolti nella linea temporale principale del file SWF host e non in quella del file SWF caricato. Di conseguenza, in alcune circostanze può verificarsi un comportamento imprevisto in fase di runtime, ad esempio nel caso in cui sia il file SWF host sia il file SWF caricato utilizzino `_root` per specificare una variabile.

In Flash Player 7 e nelle versioni successive è possibile utilizzare la proprietà `_lockroot` (`MovieClip._lockroot` property) per imporre la risoluzione dei riferimenti a `_root` eseguiti da un clip filmato alla relativa linea temporale, anziché alla linea temporale del file SWF che contiene quel clip filmato. Per ulteriori informazioni, vedere “[Impostazione di una linea temporale principale per i file SWF caricati](#)” a pagina 386. Per ulteriori informazioni sull'uso di `_root` e `_lockroot`, consultare il Capitolo 19, “[Procedure ottimali e convenzioni di codifica per ActionScript 2.0](#)” a pagina 791.

Un file SWF può caricare un altro file SWF da qualsiasi posizione in Internet. Tuttavia, per rendere possibile l'accesso di un file SWF ai dati (variabili, metodi e così via) definiti in un altro file SWF, è necessario che i due file abbiano origine dallo stesso dominio. In Flash Player 7 e versioni successive, gli script tra più domini non sono consentiti, eccetto nel caso in cui il file SWF caricato fornisca una diversa indicazione chiamando `System.security.allowDomain()`.

Per ulteriori informazioni su `System.security.allowDomain`, vedere `allowDomain` (`security.allowDomain` method) nella *Guida di riferimento di ActionScript 2.0* e “[Informazioni su domini, sicurezza tra domini e file SWF](#)” a pagina 753.

## Informazioni sul caricamento e l'uso di file MP3 esterni

Per caricare i file MP3 in fase di runtime, utilizzare il metodo `loadSound()` della classe `Sound`. Creare in primo luogo un oggetto `Sound`, come nell'esempio seguente:

```
var song1_sound:Sound = new Sound();
```

Usare il nuovo oggetto per chiamare `loadSound()` in modo da caricare un evento o audio in streaming. I suoni associati agli eventi vengono caricati completamente prima di essere riprodotti, mentre l'audio in streaming viene riprodotto durante lo scaricamento. È possibile impostare il parametro `isStreaming` del metodo `loadSound()` per specificare se si tratta di un suono associato a un evento o di audio in streaming. Dopo aver caricato un evento audio, è necessario chiamare il metodo `start()` della classe `Sound` per riprodurre l'audio. La riproduzione dell'audio in streaming inizia quando sono stati caricati dati sufficienti nel file SWF; non è necessario utilizzare il metodo `start()`.

Ad esempio, il codice seguente consente di creare un oggetto `Sound` denominato `my_sound` e di caricare quindi un file MP3 denominato `song1.mp3`. Immettere il codice seguente nel fotogramma 1 della linea temporale:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
 true);
```

Nella maggior parte dei casi, è necessario impostare il parametro `isStreaming` su `true`, in particolare se si caricano file audio di grandi dimensioni di cui si desidera iniziare la riproduzione in tempi molto brevi, ad esempio in caso di creazione di un'applicazione "jukebox" MP3. Tuttavia, se si scaricano clip audio di dimensioni più ridotte e si desidera riprodurli in un momento specifico (ad esempio, quando un utente fa clic su un pulsante), è necessario impostare `isStreaming` su `false`.

Per stabilire il momento in cui un file audio è stato completamente scaricato, utilizzare il gestore di eventi `Sound.onLoad`. Questo gestore di eventi riceve automaticamente un valore booleano (`true` o `false`) che indica se lo scaricamento del file è riuscito.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “Caricamento di un file MP3” a pagina 654
- “Precaricamento di file MP3” a pagina 655
- “Lettura dei tag ID3 nei file MP3” a pagina 656

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio che carica file MP3, `jukebox.fla`. Questo esempio illustra come creare un jukebox utilizzando i tipi di dati, i principi di codificazione generali e diversi componenti:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox*.

## Caricamento di un file MP3

si supponga di voler creare un gioco in linea che prevede tipi di audio diversi in base al livello raggiunto dall'utente. Il codice seguente consente di caricare un file MP3 (song2.mp3) nell'oggetto Sound denominato game\_Sound e di riprodurre quindi l'audio al termine dello scaricamento:

### Per caricare un file MP3:

1. Creare un nuovo file FLA denominato **loadMP3.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var game_sound:Sound = new Sound();
game_sound.onLoad = function(success:Boolean):Void {
 if (success) {
 trace("Sound Loaded");
 game_sound.start();
 }
};
game_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3"
 false);'
```

3. Selezionare Controllo > Prova filmato per provare il file audio.

Flash Player supporta solo i tipi di file audio MP3 per il caricamento dei file audio in fase di runtime.

Per ulteriori informazioni, vedere `Sound.loadSound()`, `Sound.start()` e `Sound.onLoad` nella *Guida di riferimento di ActionScript 2.0*. Per informazioni sul precaricamento dei file MP3, vedere “[Precaricamento di file MP3](#)” a pagina 655. Per informazioni sulla creazione di un'animazione per la barra di avanzamento quando si carica un file audio, vedere “[Creazione di una barra di avanzamento per caricare file MP3 con ActionScript](#)” a pagina 684.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio che carica file MP3, jukebox.fla. Questo esempio illustra come creare un jukebox utilizzando i tipi di dati, i principi di codificazione generali e diversi componenti:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox*.

# Precaricamento di file MP3

Per precaricare file MP3, è possibile usare la funzione `setInterval()` e creare un *meccanismo di polling* che esegua una verifica dei byte caricati per un oggetto Sound o NetStream a intervalli prestabiliti. Per tenere traccia dell'avanzamento dello scaricamento dei file MP3, utilizzare i metodi `Sound.getBytesLoaded()` e `Sound.getBytesTotal()`.

Il seguente esempio utilizza `setInterval()` per verificare i byte caricati per un oggetto Sound a intervalli prestabiliti.

## Per precaricare un file MP3:

1. Creare un nuovo file FLA denominato **preloadMP3.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
// Crea un nuovo oggetto Sound per riprodurre l'audio.
var songTrack:Sound = new Sound();
// Crea una funzione di polling che tiene traccia dello stato di
// scaricamento.
// Questa è la funzione oggetto del "polling". Verifica
// l'avanzamento dello scaricamento dell'oggetto Sound passato come
// riferimento.
function checkProgress (soundObj:Object):Void {
 var numBytesLoaded:Number = soundObj.getBytesLoaded();
 var numBytesTotal:Number = soundObj.getBytesTotal();
 var numPercentLoaded:Number = Math.floor(numBytesLoaded /
 numBytesTotal * 100);
 if (!isNaN(numPercentLoaded)) {
 trace(numPercentLoaded + "% loaded.");
 }
};
// Al termine del caricamento del file, annulla il polling
// dell'intervalllo.
songTrack.onLoad = function ():Void {
 trace("load complete");
 clearInterval(poll);
};
// Carica il file MP3 in streaming e chiama checkProgress(),
songTrack.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
 true);
var poll:Number = setInterval(checkProgress, 100, songTrack);
```

3. Selezionare Controllo > Prova filmato per provare il file audio.

Il pannello Output mostra lo stato del caricamento.

È possibile usare la stessa tecnica di polling per precaricare file FLV esterni. Per determinare il numero totale e il numero corrente di byte caricati per un file FLV, utilizzare le proprietà `NetStream.bytesLoaded` e `NetStream.bytesTotal` (per ulteriori informazioni, vedere `bytesLoaded` (`NetStream.bytesLoaded` property) e `bytesTotal` (`NetStream.bytesTotal` property)).

Per ulteriori informazioni, vedere `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `setInterval()`, `Sound.getBytesLoaded()` e `Sound.getBytesTotal()` nella *Guida di riferimento di ActionScript 2.0*.

Per informazioni sulla creazione di un'animazione per la barra di avanzamento, vedere [“Creazione di una barra di avanzamento per caricare file MP3 con ActionScript” a pagina 684](#).

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio che carica file MP3, jukebox.fla. Questo esempio illustra come creare un jukebox utilizzando i tipi di dati, i principi di codificazione generali e diversi componenti:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox*.

## Lettura dei tag ID3 nei file MP3

I tag ID3 sono campi di dati che vengono aggiunti a un file MP3. I tag ID3 contengono le informazioni sul file, come il nome di una canzone, l'album e l'artista.

Per leggere i tag ID3 da un file MP3, utilizzare la proprietà `Sound.ID3`, le cui proprietà corrispondono ai nomi dei tag ID3 inclusi nel file MP3 in fase di caricamento. Per determinare quando sono disponibili i tag ID3 per un file MP3 in fase di scaricamento, utilizzare il gestore di eventi `Sound.onID3`. Flash Player7 supporta i tag versione 1.0, 1.1, 2.3 e 2.4; non sono supportati i tag versione 2.2.

L'esempio seguente carica un file MP3 denominato song1.mp3 nell'oggetto `Sound song_sound`. Quando sono disponibili i tag ID3 per il file, il nome dell'artista e il titolo del brano vengono visualizzati in un campo di testo denominato `display_txt`.

### **Per leggere i tag ID3 da un file MP3:**

1. Creare un nuovo file FLA denominato **id3.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
this.createTextField("display_txt", this.getNextHighestDepth(), 0, 0,
100, 100);
display_txt.autoSize = "left";
display_txt.multiline = true;
var song_sound:Sound = new Sound();
song_sound.onLoad = function() {
 song_sound.start();
};
song_sound.onID3 = function():Void {
 display_txt.text += "Artist:\t" + song_sound.id3.artist + "\n";
 display_txt.text += "Song:\t" + song_sound.id3.songname + "\n";
};
song_sound.loadSound("http://www.helpexamples.com/flash/sound/
song1.mp3");
```

3. Selezionare Controllo > Prova filmato per provare il file audio.

I tag ID3 appaiono sullo stage durante la riproduzione del file audio.

I tag ID3 2.0 si trovano all'inizio dei file MP3 (prima dei dati audio); pertanto, sono disponibili non appena comincia lo scaricamento del file. I tag ID3 1.0 si trovano invece alla fine del file (dopo i dati audio) e non sono quindi disponibili fino a quando non viene completato lo scaricamento dell'intero file MP3.

Il gestore di eventi `onID3` viene richiamato ogni volta che sono disponibili nuovi dati ID3. Di conseguenza, se un file MP3 contiene tag ID3 2.0 e ID3 1.0, il gestore `onID3` viene chiamato due volte, in quanto i tag si trovano in punti diversi del file.

Per un elenco dei tag ID3 supportati, vedere `id3` (`Sound.id3` property) nella *Guida di riferimento di ActionScript 2.0*.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio che carica file MP3, `jukebox.fla`. Questo esempio illustra come creare un jukebox utilizzando i tipi di dati, i principi di codificazione generali e diversi componenti:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox*.

# Assegnazione di identificatori di concatenamento alle risorse della libreria

È possibile assegnare identificatori di concatenamento alle risorse della libreria, come clip filmato e simboli di caratteri. In Flash Basic 8 e Flash Professional 8, è possibile impostare gli identificatori di concatenamento sulle risorse audio e di immagine della libreria. In questo modo si supporta l'impiego di file di immagine e audio con le librerie condivise e con la nuova classe `BitmapData`.

L'esempio seguente aggiunge un'immagine bitmap alla libreria con un indicatore di concatenamento impostato su `myImage`. Quindi l'immagine viene aggiunta allo stage e resa trascinabile.

## Per utilizzare gli indicatori di concatenamento con i file bitmap:

1. Creare un nuovo file FLA denominato `linkBitmap.fla`.
2. Importare un'immagine bitmap nella libreria.
3. Fare clic con il pulsante destro del mouse (Windows) o premere Ctrl e fare clic (Macintosh) sull'immagine presente nella libreria, quindi selezionare Concatenamento dal menu di scelta rapida.
4. Selezionare Esporta per ActionScript ed Esporta nel primo fotogramma e digitare `myImage` nella casella di testo Identificatore.
5. Fare clic su OK per impostare l'identificatore di concatenamento.
6. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
import flash.display.BitmapData;
// Crea imageBmp e associa la bitmap della libreria.
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
// crea il clip filmato e associa imageBmp
this.createEmptyMovieClip("imageClip", 10);
imageClip.attachBitmap(imageBmp, 2);
// rende il clip trascinabile
imageClip.onPress = function() {
 this.startDrag();
};
imageClip.onRelease = function() {
 this.stopDrag();
}
```

7. Selezionare Controllo > Prova filmato per provare il documento.

La bitmap della libreria appare sullo stage e l'immagine è trascinabile.

# Informazioni sull'uso dei file video FLV

Il formato di file FLV contiene dati audio e video codificati per la distribuzione attraverso Flash Player. Per esempio, se si dispone di un file video QuickTime o Windows Media, si utilizza un codificatore (come Flash 8 Video Encoder o Sorense Squeeze) per convertire tale file in un file FLV.

Flash Player 7 supporta i file FLV codificati con il codec per video Sorenson Spark. Flash Player 8 supporta i file FLV codificati con Sorenson Spark o On2 VP6 in Flash Professional 8. Il codec per video On2 VP6 supporta un canale alfa. Versioni diverse di Flash Player supportano i file FLV in modi differenti. Per ulteriori informazioni, consultare la tabella seguente:

| <b>Codec</b>   | <b>Versione file SWF<br/>(versione di pubblicazione)</b> | <b>Versione Flash Player richiesta per la riproduzione</b> |
|----------------|----------------------------------------------------------|------------------------------------------------------------|
| Sorenson Spark | 6                                                        | 6, 7 o 8.                                                  |
|                | 7                                                        | 7, 8                                                       |
| On2 VP6        | 6                                                        | 8*                                                         |
|                | 7                                                        | 8                                                          |
|                | 8                                                        | 8                                                          |

\* Se il file SWF carica un file FLV, è possibile utilizzare il video On2 VP6 con l'obbligo di ripubblicare il file SWF per Flash Player 8, sempre che gli utenti, per visualizzare il file SWF, utilizzino Flash Player 8. Solo Flash Player 8 supporta sia la pubblicazione che la riproduzione di video On2 VP6.

Per informazioni sui concetti fondamentali del video, come streaming, scaricamento progressivo, codifica, importazione e problemi di banda, vedere [Capitolo 11, “Operazioni con i file video”](#) nella guida *Uso di Flash*.

Questa sezione descrive l'utilizzo di video FLV senza componenti. È anche possibile utilizzare il componente FLVPlayback per riprodurre file FLV, oppure la classe VideoPlayback per creare un video player personalizzato che carichi i file FLV in modo dinamico (vedere [www.macromedia.com/devnet/flash](http://www.macromedia.com/devnet/flash) o [www.macromedia.com/support/documentation\\_it/](http://www.macromedia.com/support/documentation_it/)).

Per informazioni sull'utilizzo di video FLV con i componenti FLVPlayback e Media, consultare le seguenti sezioni:

- “[Componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 517
- “[Componenti Media \(solo Flash Professional\)](#)” a pagina 862

In alternativa all'importazione di video nell'ambiente di creazione Flash, è possibile utilizzare ActionScript per riprodurre dinamicamente i file FLV esterni in Flash Player. È possibile riprodurre i file FLV da un indirizzo HTTP oppure dal file system locale. Per riprodurre file FLV, utilizzare le classi NetConnection e NetStream e il metodo `attachVideo()` della classe Video. Per ulteriori informazioni, vedere `NetConnection`, `NetStream` e `attachVideo` (`Video.attachVideo` method) nella *Guida di riferimento di ActionScript 2.0*.

È possibile creare file FLV importando video nello strumento di creazione di codice di Flash ed esportandolo come file FLV. Se è installato Flash Professional, è possibile utilizzare il plug-in FLV Export per esportare i file FLV dalle applicazioni di videomontaggio supportate.

Se si utilizzano file FLV esterni, sono disponibili alcune funzionalità che invece non sono attive se si utilizzano video incorporati:

- Nei documenti Flash è possibile utilizzare video clip di lunghezza maggiore senza alcun deterioramento delle prestazioni di riproduzione. I file FLV esterni vengono riprodotti mediante la *memoria cache*. Di conseguenza, i file di grandi dimensioni sono suddivisi in piccoli file memorizzati; sono accessibili dinamicamente e richiedono una minore quantità di memoria rispetto ai file video incorporati.
- I file FLV esterni possono essere caratterizzati da una frequenza fotogrammi diversa rispetto a quella del documento Flash in cui sono riprodotti. Ad esempio, è possibile impostare la frequenza fotogrammi del documento Flash su 30 f/s e la frequenza fotogrammi video su 21 f/s. Questa impostazione offre un migliore controllo del video rispetto al video incorporato, per garantire riproduzioni più omogenee. Consente anche di riprodurre i file FLV con frequenze di fotogrammi diverse senza dover modificare il contenuto Flash esistente.
- Con i file FLV esterni, non è necessario interrompere la riproduzione del documento durante il caricamento del file video. In alcuni casi i file video importati possono interrompere la riproduzione del documento per eseguire alcune funzioni, come l'accesso a un'unità CD-ROM. I file FLV possono eseguire delle funzioni indipendentemente dal documento Flash, senza quindi interromperne la riproduzione.
- L'acquisizione di contenuto video risulta più semplice se si utilizzano file FLV esterni, perché è possibile utilizzare gestori di eventi per accedere ai metadati relativi al video.

**SUGGERIMENTO**

Per caricare file FLV da un server Web, potrebbe essere necessario registrare l'estensione e il tipo MIME del file con il server Web. A questo proposito, consultare la documentazione del server Web. Il tipo MIME dei file FLV è `video/x-flv`. Per ulteriori informazioni, vedere “[Informazioni sulla configurazione di file FLV per l'hosting su un server](#)” a pagina 679.

Per ulteriori informazioni sui video FLV, consultare i seguenti argomenti:

- “Creazione di un oggetto video” a pagina 661
- “Riproduzione dinamica di file FLV esterni” a pagina 662
- “Creazione di un banner video” a pagina 663
- “Precarica di file FLV” a pagina 665
- “Lavoro con i cue point” a pagina 667
- “Operazioni con i metadati” a pagina 677
- “Informazioni sulla configurazione di file FLV per l'hosting su un server” a pagina 679
- “Informazioni sull'indirizzamento di file FLV locali su Macintosh” a pagina 680

## Creazione di un oggetto video

Prima di caricare e manipolare un video con ActionScript, è necessario creare un oggetto video, trascinarlo sullo stage e assegnargli un nome di istanza. L'esempio seguente mostra come aggiungere un'istanza video a un'applicazione.

### Per creare un oggetto video:

1. Con un documento aperto nello strumento di creazione Flash, selezionare Nuovo Video dal menu a comparsa del pannello Libreria (Finestra > Libreria).
2. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
3. Fare clic su OK per creare un oggetto video.
4. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza dell'oggetto video.
5. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).

Ora sullo stage è presente un'istanza video, per la quale è possibile aggiungere il codice ActionScript così da poter caricare o manipolare l'istanza in diversi modi.

Per informazioni sul caricamento dinamico dei file FLV, vedere “[Riproduzione dinamica di file FLV esterni](#)”. Per informazioni sulla creazione di un banner video, vedere “[Creazione di un banner video](#)” a pagina 663.

# Riproduzione dinamica di file FLV esterni

In fase di runtime è possibile caricare i file FLV che devono essere riprodotti in un file SWF. È possibile caricarli in un oggetto video o in un componente come FLVPlayback. L'esempio seguente mostra come riprodurre un file denominato clouds.flv in un oggetto video.

## Per riprodurre un file FLV esterno in un documento Flash:

1. Creare un nuovo documento Flash denominato playFLV.fla.
2. Nel pannello Libreria (Finestra > Libreria), scegliere Nuovo Video dal menu a comparsa.
3. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
4. Fare clic su OK per creare un oggetto video.
5. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza dell'oggetto video.
6. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
7. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).

### 8. Immettere il codice seguente nel pannello Azioni:

```
this.createTextField("status_txt", 999, 0, 0, 100, 100);
status_txt.autoSize = "left";
status_txt.multiline = true;
// Crea un oggetto NetConnection
var my_nc:NetConnection = new NetConnection();
// Crea una connessione in streaming locale
my_nc.connect(null);
// Crea un oggetto NetStream e definisce una funzione onStatus()
var my_ns:NetStream = new NetStream(my_nc);
my_ns.onStatus = function(infoObject:Object):Void {
 status_txt.text += "status (" + this.time + " seconds)\n";
 status_txt.text += "\tLevel: " + infoObject.level + "\n";
 status_txt.text += "\tCode: " + infoObject.code + "\n\n";
};
// Associa la sorgente video NetStream all'oggetto Video
my_video.attachVideo(my_ns);
// Imposta il tempo di buffer
my_ns.setBufferTime(5);
// Inizia la riproduzione del file FLV
my_ns.play("http://www.helpexamples.com/flash/video/clouds.flv");
```

9. Selezionare Controllo > Prova filmato per provare il documento.

Per informazioni sul caricamento dei file FLV, vedere “[Precaricamento di file FLV](#)” a pagina 507. Per informazioni sul caricamento dinamico di video FLV nei componenti, vedere “[Creazione di un'applicazione con il componente FLVPlayback](#)” a pagina 519. Per informazioni sui file FLV e il server, e i file FLV e la loro riproduzione in locale su Macintosh, vedere “[Informazioni sulla configurazione di file FLV per l'hosting su un server](#)” a pagina 679.

## Creazione di un banner video

Il contenuto video all'interno dei banner e di altre pubblicità Flash viene spesso utilizzato per promozione, come la visualizzazione di anteprime di filmati Flash o di pubblicità televisive. L'esempio seguente mostra come creare un'istanza video e aggiungere il codice ActionScript a un file FLA per generare un banner pubblicitario contenente un video.

### Per creare un banner video:

1. Creare un nuovo documento Flash denominato **vidBanner.fla**.
2. Selezionare Elabora > Documento.
3. Modificare le dimensioni del file FLA, digitare **468** nella casella di testo della larghezza e **60** in quella dell'altezza.
4. Nel pannello Libreria (Finestra > Libreria), scegliere Nuovo Video dal menu a comparsa.
5. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
6. Fare clic su OK per creare un oggetto video.
7. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza video.
8. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
9. Con l'istanza video ancora selezionata, digitare **105** nella casella di testo della larghezza e **60** in quella dell'altezza della finestra di ispezione Proprietà.
10. Trascinare l'istanza video in un punto dello stage o utilizzare la finestra di ispezione Proprietà per impostare le sue coordinate *x* e *y*.
11. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).

**12.** Aggiungere il codice seguente nel pannello Azioni:

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.setBufferTime(5);
my_ns.play("http://www.helpexamples.com/flash/video/vbanner.flv");
```

**13.** Selezionare Inserisci > Linea temporale > Livello per creare un nuovo livello e chiamarlo **button**.

**14.** Nel pannello Strumenti, selezionare lo strumento Rettangolo.

**15.** Nella sezione Colori del pannello Strumenti, fare clic sull'icona della matita per selezionare il controllo Colore tratto.

**16.** Selezionare Nessun colore per disattivare il contorno del rettangolo.

**17.** Trascinare diagonalmente il puntatore sullo stage per creare un rettangolo.

Le dimensioni del rettangolo non sono importanti poiché verrà ridimensionato attraverso la finestra di ispezione Proprietà.

**18.** Fare clic sullo strumento Selezione nel pannello Strumenti e poi sul rettangolo presente nello stage per selezionarlo.

**19.** Con il rettangolo ancora selezionato, digitare **468** nella casella di testo della larghezza e **60** in quella dell'altezza della finestra di ispezione Proprietà. Impostare quindi le coordinate X e Y (le caselle di testo X e Y) su **0**.

**20.** Con il rettangolo selezionato sullo stage, premere F8 per trasformarlo in un simbolo.

**21.** Nella finestra di dialogo Converti in simbolo, digitare **invisible btn** nella casella Nome, selezionare Pulsante e poi fare clic su OK.

**22.** Fare doppio clic sul nuovo pulsante nello stage per entrare nella modalità di modifica del simbolo.

Al momento il rettangolo si trova sul primo fotogramma Su del pulsante creato. Questo è lo stato Su del pulsante: ciò che l'utente vede quando il pulsante si trova sullo stage.

Tuttavia, si desidera che il pulsante non sia visibile sullo stage, quindi è necessario spostare il rettangolo sul fotogramma Area attiva, cioè l'area attiva del pulsante (la zona attiva su cui un utente può fare clic per attivare le azioni del pulsante).

**23.** Fare clic sul fotogramma chiave del fotogramma Su e tenere premuto il pulsante del mouse mentre si trascina il fotogramma chiave sul fotogramma Area attiva.

Ora è possibile fare clic sull'intera area del banner, ma il pulsante non viene rappresentato visivamente.

**24.**Fare clic su Scena 1 per tornare alla linea temporale principale.

Sull'area del banner appare un rettangolo verdastro che rappresenta l'area attiva del pulsante invisibile.

**25.**Selezionare il pulsante creato, aprire la finestra di ispezione Proprietà e digitare **inv\_btn** nella casella di testo Nome istanza.

**26.**Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
inv_btn.onRelease = function(){
 getURL("http://www.macromedia.com");
};
```

**27.**Eseguire altre modifiche al banner, come l'aggiunta di immagini o testo.

**28.**Selezionare Controllo > Prova filmato per provare il banner in Flash Player.

In questo esempio è stato creato un banner le cui dimensioni sono state impostate su quelle standard stabilite dall'Interactive Advertising Bureau. Per informazioni sulle dimensioni standard delle pubblicità (e altre utili istruzioni), consultare la pagina Interactive Advertising Bureau's Standards and Guidelines all'indirizzo [www.iab.net/standards/adunits.asp](http://www.iab.net/standards/adunits.asp).

Nonostante le istruzioni standardizzate, assicurarsi di confermare innanzitutto le linee guida per la pubblicità con il fornitore, il cliente o il sito Web con cui si sta lavorando. Se si invia il banner a un'agenzia pubblicitaria, assicurarsi che i file abbia dimensioni specifiche, sia indirizzato a una versione di Flash Player e la frequenza dei fotogrammi sia quella stabilita. Inoltre, è necessario prendere in considerazione le regole riguardanti i tipi di media che si possono utilizzare, il codice del pulsante impiegato nel file FLA e così via.

## Precarica di file FLV

Per controllare lo stato dello scaricamento dei file FLV, utilizzare le proprietà

`NetStream.bytesLoaded` e `NetStream.bytesTotal`. Per ottenere il numero totale e il

numero corrente di byte caricati per un file FLV, utilizzare le proprietà

`NetStream.bytesLoaded` e `NetStream.bytesTotal`.

L'esempio seguente utilizza le proprietà `bytesLoaded` e `bytesTotal` che mostrano lo stato del caricamento del file video1.flv nell'istanza dell'oggetto video denominata `my_video`. Viene creato dinamicamente un campo di testo, denominato `loaded_txt`, per visualizzare le informazioni sull'avanzamento del caricamento.

### **Per caricare un file FLV:**

1. Creare un nuovo file FLA denominato **preloadFLV.fla**.
2. Nel pannello Libreria (Finestra > Libreria), scegliere Nuovo Video dal menu a comparsa.
3. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
4. Fare clic su OK per creare un oggetto video.
5. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza dell'oggetto video.
6. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
7. Con l'istanza video ancora selezionata, digitare **320** nella casella di testo della larghezza e **213** in quella dell'altezza della finestra di ispezione Proprietà.
8. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
9. Immettere il codice seguente nel pannello Azioni:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
 lights_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
 160, 22);
var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
 stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
 var pctLoaded:Number = Math.round(my_ns.bytesLoaded / my_ns.bytesTotal
 * 100);
 loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
 Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
 "%)";
 progressBar_mc.bar_mc._xscale = pctLoaded;
 if (pctLoaded >= 100) {
 clearInterval(loaded_interval);
 }
}
```

## 10. Selezionare Controllo > Prova filmato per provare il codice.

**NOTA**

Se la barra di avanzamento viene caricata subito, il video è stato memorizzato nella cache del disco rigido (o perché l'esempio è stato provato o perché è stato caricato in una procedura diversa). Se si verifica questa situazione, inserire un file FLV nel server e caricarlo.

Per precaricare file FLV è inoltre possibile utilizzare il metodo `NetStream.setBufferTime()`. Tale metodo acquisisce un solo parametro che indica il numero di secondi di streaming del file FLV da inserire nel buffer prima che venga avviata la riproduzione. Per ulteriori informazioni, vedere `setBufferTime (NetStream.setBufferTime method)`, `getBytesLoaded (MovieClip.getBytesLoaded method)`, `getBytesTotal (MovieClip.getBytesTotal method)`, `bytesLoaded (NetStream.bytesLoaded property)`, `bytesTotal (NetStream.bytesTotal property)` e `setInterval function` nella *Guida di riferimento di ActionScript 2.0*.

## Lavoro con i cue point

Con Flash Video è possibile utilizzare diversi tipi di cue point. È possibile utilizzare ActionScript per interagire con i cue point incorporati in un file FLV (quando si crea il file FLV) o generati attraverso ActionScript.

**Cue point di navigazione** Quando si codifica il file FLV i cue point di navigazione vengono incorporati nel flusso FLV e nel pacchetto di metadati FLV. Si utilizzano i cue point di navigazione per consentire agli utenti di eseguire una ricerca in una specifica parte di un file.

**Cue point di evento** Quando si codifica il file FLV i cue point di evento vengono incorporati nel flusso FLV e nel pacchetto di metadati FLV. È possibile scrivere il codice per gestire gli eventi che vengono attivati in punti specifici durante la riproduzione FLV.

**Cue point ActionScript** Cue point esterni creati utilizzando codice ActionScript. È possibile scrivere il codice che attiva questi cue point in relazione alla riproduzione del video. Questi cue point sono meno accurati rispetto a quelli incorporati (fino a un decimo di secondo), perché il video player li traccia separatamente.

I cue point di navigazione creano un fotogramma chiave nella posizione del cue point specificata, quindi è possibile utilizzare il codice per spostare l'indicatore di riproduzione di un video player in quella posizione. In un file FLV è possibile impostare punti particolari in cui gli utenti possono eseguire una ricerca. Per esempio, il video può essere composto da più capitoli o segmenti e lo si può controllare incorporando i cue point di navigazione nel file.

Se si prevede di creare un'applicazione in cui si desidera che gli utenti si spostino su un cue point, occorre realizzare e incorporare i cue point quando si codifica il file invece di utilizzare i cue point ActionScript. Occorre incorporare i cue point nel file FLV, perché si può lavorare in modo più preciso. Per ulteriori informazioni sulla codifica dei file FLV con i cue point, vedere [“Incorporamento dei cue point \(solo Flash Professional\)”](#) a pagina 337 nella guida *Uso di Flash*.

È possibile accedere ai parametri dei cue point scrivendo codice ActionScript. I parametri dei cue point fanno parte dell'oggetto evento ricevuto con l'evento `cuePoint` (`event.info.parameters`).

## Tracciamento dei cue point da un file FLV

È possibile tracciare i cue point incorporati in un documento FLV attraverso `NetStream.onMetaData`. Per visualizzare le informazioni sui cue point è necessario eseguire la recursione sui metadati che vengono restituiti.

Il codice seguente traccia i cue point di un file FLV:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.onMetaData = function(metaProp:Object) {
 trace("The metadata:");
 traceMeta(metaProp);
 // traceObject(metaProp, 0);
};

my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

function traceMeta(metaProp:Object):Void {
 var p:String;
 for (p in metaProp) {
 switch (p) {
 case "cuePoints" :
 trace("cuePoints: ");
 //cycles through the cue points
 var cuePointArr:Array = metaProp[p];
 for (var j:Number = 0; j < cuePointArr.length; j++) {
 //cycle through the current cue point parameters
 trace("\t cuePoints[" + j + "]:");
 var currentCuePoint:Object = metaProp[p][j];
 var metaPropPjParams:Object = currentCuePoint.parameters;
 trace("\t\t name: " + currentCuePoint.name);
 trace("\t\t time: " + currentCuePoint.time);
 trace("\t\t type: " + currentCuePoint.type);
 if (metaPropPjParams != undefined) {
 trace("\t\t parameters:");
 }
 }
 }
 }
}
```

```

 traceObject(metaPropPJParams, 4);
 }
}
break;
default :
 trace(p + ": " + metaProp[p]);
 break;
}
}
}

function traceObject(obj:Object, indent:Number):Void {
 var indentString:String = "";
 for (var j:Number = 0; j < indent; j++) {
 indentString += "\t";
 }
 for (var i:String in obj) {
 if (typeof(obj[i]) == "object") {
 trace(indentString + " " + i + ": [Object]");
 traceObject(obj[i], indent + 1);
 } else {
 trace(indentString + " " + i + ": " + obj[i]);
 }
 }
}
}

```

Appare l'output seguente:

```

The metadata:
canSeekToEnd: true
cuePoints:
 cuePoints[0]:
 name: point1
 time: 0.418
 type: navigation
 parameters:
 lights: beginning
 cuePoints[1]:
 name: point2
 time: 7.748
 type: navigation
 parameters:
 lights: middle
 cuePoints[2]:
 name: point3
 time: 16.02
 type: navigation
 parameters:
 lights: end
audiocodecid: 2
audiodelay: 0.038
audiodatarate: 96

```

```
videocodecid: 4
framerate: 15
videodatarate: 400
height: 213
width: 320
duration: 16.334
```

Per informazioni sull'utilizzo dei cue point con il componente FLVPlayback, vedere ["Utilizzo dei cue point incorporati con il componente FLVPlayback \(solo Flash Professional\)"](#).

## Utilizzo dei cue point incorporati con il componente FLVPlayback (solo Flash Professional)

Quando si utilizza il componente FLVPlayback, nella finestra di ispezione Proprietà è possibile visualizzare i cue point di un file FLV. Dopo aver impostato la proprietà contentPath per l'istanza FLVPlayback, è possibile visualizzare qualsiasi cue point incorporato nel file video. Utilizzando la scheda Parametri, trovare la proprietà cuePoints e fare clic sull'icona della lente d'ingrandimento per visualizzare un elenco dei cue point presenti nel file.



Per visualizzare i cue point nella scheda Parametri è necessario digitare il nome del file FLV nella casella di testo contentPath invece di utilizzare il codice per assegnare contentPath.

L'esempio seguente mostra come utilizzare le informazioni sui cue point con il componente FLVPlayback.

### Per utilizzare i cue point con il componente FLVPlayback:

1. Creare un nuovo documento Flash denominato **cueFlv.fla**.
2. Aprire il pannello Componenti (Finestra > Componenti) e trascinare sullo stage un'istanza dei componenti FLVPlayback e TextArea.
3. Nella finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà), selezionare il componente TextArea e digitare **my\_ta** nella casella di testo Nome istanza.
4. Con il componente TextArea ancora selezionato, digitare **200** nella casella di testo della larghezza e **100** in quella dell'altezza.
5. Selezionare l'istanza FLVPlayback sullo stage e digitare **my\_flvPb** nella casella di testo Nome istanza.

- 6.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var my_flvPb:mx.video.FLVPlayback;
var my_ta:mx.controls.TextArea;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
 cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
 my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
 "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

- 7.** Selezionare Controllo > Prova filmato per provare il file SWF.

Il tempo trascorso appare nell'istanza TextArea quando l'indicatore di riproduzione passa su ciascun cue point incorporato nel documento.

Per ulteriori informazioni sul lavoro con il componente FLVPlayback, vedere “[Componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 517.

## Creazione di cue point con ActionScript per l'utilizzo con componenti (solo Flash Professional)

È possibile creare i cue point con ActionScript e poi utilizzarli con un'istanza dell'oggetto video o con uno dei componenti del video player (FLVPlayback per Flash Player 8 o MediaPlayback per Flash Player 7). Gli esempi seguenti mostrano la facilità di utilizzo del codice ActionScript per creare i cue point e l'impiego di uno script per accedere a essi.

### NOTA

Se si desidera aggiungere a un'applicazione le funzioni di navigazione è necessario includere i cue point di navigazione in un documento. Per ulteriori informazioni, vedere “[Lavoro con i cue point](#)” a pagina 667. Per un esempio di lavoro con i cue point incorporati, vedere “[Utilizzo dei cue point incorporati con il componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 670.

### Per creare e utilizzare i cue point con il componente FLVPlayback:

1. Creare un nuovo documento Flash denominato **cueFlvPb.fla**.
2. Trascinare sullo stage un'istanza del componente FLVPlayback dal pannello Componenti (Finestra > Componenti).  
Il componente si trova nella cartella FLVPlayback - Player 8.
3. Selezionare il componente e aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
4. Digitare **my\_flvPb** nella casella di testo Nome istanza.
5. Trascinare un'istanza del componente TextArea dal pannello Componenti allo stage.

6. Selezionare il componente TextArea e digitare **my\_ta** nella casella di testo Nome istanza.
7. Con il componente TextArea ancora selezionato, digitare **200** nella casella di testo della larghezza e **100** in quella dell'altezza.
8. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var my_flvPb:mx.video.FLVPlayback;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
clouds.flv";

// Crea un oggetto cuePoint
var cuePt:Object = new Object();
cuePt.time = 1;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
// Aggiunge un cue point AS.
my_flvPb.addASCuePoint(cuePt);

// Aggiunge un altro cue point AS.
my_flvPb.addASCuePoint(2, "elapsed_time2");

// Visualizza le informazioni sui cue point nel campo di testo.
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject) {
 my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
 "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

9. Selezionare Controllo > Prova filmato per provare il codice.

I cue point seguenti vengono tracciati nel pannello Output:

```
Elapsed time in seconds: 1.034
Elapsed time in seconds: 2.102
```

Per informazioni su `addASCuePoint()`, vedere “[FLVPlayback.addASCuePoint\(\)](#)” a pagina 568. Per informazioni sul lavoro con i cue point e il componente FLVPlayback, vedere “[Uso dei cue point](#)” a pagina 526 e “[Componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 517.

L'esempio seguente mostra come aggiungere i cue point in fase di runtime e poi tracciarli quando un file FLV viene riprodotto nel componente MediaPlayback.

**Per creare e utilizzare i cue point con il componente MediaPlayback:**

1. Creare un nuovo documento Flash denominato cuePointMP.fla.
2. Trascinare sullo stage un'istanza del componente MediaPlayback dal pannello Componenti (Finestra > Componenti).  
Il componente si trova nella cartella Media - Player 6 - 7.
3. Selezionare il componente e aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
4. Digitare **my\_mp** nella casella di testo Nome istanza.
5. Selezionare la scheda Parametri e fare clic su Avvia la finestra di ispezione dei componenti.
6. Nella finestra di ispezione dei componenti, digitare <http://www.helpexamples.com/flash/video/clouds.flv> nella casella di testo URL.
7. Aprire il pannello Azioni (Finestra > Azioni) e immettere il seguente codice nel pannello Script:

```
import mx.controls.MediaPlayback;
var my_mp:MediaPlayback;
my_mp.autoPlay = false;
my_mp.addEventListener("cuePoint", doCuePoint);
my_mp.addCuePoint("one", 1);
my_mp.addCuePoint("two", 2);
my_mp.addCuePoint("three", 3);
my_mp.addCuePoint("four", 4);
function doCuePoint(eventObj:Object):Void {
 trace(eventObj.type + " = {cuePointName:" + eventObj.cuePointName +
 " cuePointTime:" + eventObj.cuePointTime + "}");
}
```

8. Selezionare Controllo > Prova filmato per provare il codice.

I cue point seguenti vengono tracciati nel pannello Output:

```
cuePoint = {cuePointName:one cuePointTime:1}
cuePoint = {cuePointName:two cuePointTime:2}
cuePoint = {cuePointName:three cuePointTime:3}
cuePoint = {cuePointName:four cuePointTime:4}
```

Per ulteriori informazioni sul lavoro con il componente MediaPlayback, vedere “[Componenti Media \(solo Flash Professional\)](#)” a pagina 862. Per ulteriori informazioni sul lavoro con il componente FLVPlayback, vedere “[Componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 517.

## Aggiunta della funzionalità di ricerca con i cue point (solo Flash Professional)

È possibile incorporare i cue point di navigazione in un file FLV per aggiungere alle applicazioni la funzionalità di ricerca. Il metodo `seekToNavCuePoint()` del componente FLVPlayback localizza il cue point presente nel file FLV con il nome specifico nel momento indicato o successivamente. È possibile specificare un nome come stringa (come "part1" o "theParty").

È anche possibile utilizzare il metodo `seekToNextNavCuePoint()`, che ricerca il cue point di navigazione successivo, in base alla proprietà `playheadTime` corrente. È possibile passare il metodo come parametro, `time`, che rappresenta il punto iniziale dal quale cominciare la ricerca del cue point di navigazione successivo. Il valore predefinito è la proprietà `playheadTime` corrente.

In alternativa, è anche possibile cercare una durata specifica del file FLV, utilizzando il metodo `seek()`.

Negli esempi seguenti, si aggiunge un pulsante utilizzato per spostarsi tra i cue point o a una durata specifica in un file FLV che viene riprodotto nel componente FLVPlayback, e un pulsante per raggiungere un cue point particolare.

### Per cercare una durata specifica:

1. Creare un nuovo documento Flash denominato **seekduration.fla**.
2. Trascinare un'istanza del componente FLVPlayback dal pannello Componenti (Finestra > Componenti).  
Il componente si trova nella cartella FLVPlayback - Player 8.
3. Selezionare il componente e aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
4. Digitare **my\_flvPb** nella casella di testo Nome istanza.
5. Trascinare un'istanza del componente Button dal pannello Componenti allo stage.
6. Selezionare il componente Button e digitare **my\_button** nella casella di testo Nome istanza.

- 7.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
import mx.controls.Button;
import mx.video.FLVPlayback;
var seek_button:Button;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
 sheep.flv";
seek_button.label = "Seek";
seek_button.addEventListener("click", seekFlv);
function seekFlv(eventObj:Object):Void {
 // ricerca a 2 secondi
 my_flvPb.seek(2);
}
```

- 8.** Selezionare Controllo > Prova filmato per provare il codice.

Quando si fa clic sul pulsante, l'indicatore di riproduzione del video si sposta sulla durata specificata. 2 secondi all'interno del video.

**Per aggiungere la funzionalità di ricerca con il componente FLVPlayback:**

- 1.** Creare un nuovo documento Flash denominato **seek1.fla**.
- 2.** Trascinare un'istanza del componente FLVPlayback dal pannello Componenti (Finestra > Componenti).  
Il componente si trova nella cartella FLVPlayback - Player 8.
- 3.** Selezionare il componente e aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
- 4.** Digitare **my\_flvPb** nella casella di testo Nome istanza.
- 5.** Trascinare un'istanza del componente Button dal pannello Componenti allo stage.
- 6.** Selezionare il componente Button e digitare **my\_button** nella casella di testo Nome istanza.
- 7.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
 cuepoints.flv";
my_button.label = "Next cue point";

function clickMe(){
 my_flvPb.seekToNextNavCuePoint();
}
my_button.addEventListener("click", clickMe);
```

**8.** Selezionare Controllo > Prova filmato per provare il codice.

Il file cuepoints.flv contiene tre cue point di navigazione: uno quasi all'inizio, uno a metà e uno alla fine del file video. Quando si fa clic sul pulsante, l'istanza FLVPlayback cerca il cue point successivo finché raggiunge l'ultimo presente nel file video.

In un file FLV, è anche possibile cercare un cue point specifico utilizzando il metodo seekToCuePoint(), come mostrato nell'esempio seguente.

**Per cercare uno specifico cue point:**

**1.** Creare un nuovo documento Flash denominato **seek2.fla**.

**2.** Trascinare un'istanza del componente FLVPlayback dal pannello Componenti (Finestra > Componenti).

Il componente si trova nella cartella FLVPlayback - Player 8.

**3.** Selezionare il componente e aprire la finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).

**4.** Digitare **my\_flvPb** nella casella di testo Nome istanza.

**5.** Con l'istanza FLVPlayback ancora selezionata, fare clic sulla scheda Parametri.

**6.** Digitare <http://www.helpexamples.com/flash/video/cuepoints.flv> nella casella di testo contentPath.

Quando si immette l'URL nella casella di testo contentPath, i cue point appaiono nella scheda Parametri (vicino al parametro cuePoint). Di conseguenza, è possibile stabilire il nome del cue point che si desidera trovare nel codice. Se si fa clic sull'icona della lente d'ingrandimento, è possibile visualizzare tutti i cue point del file video e le informazioni relative a ciascuno di essi in una tabella.

**7.** Trascinare un'istanza del componente Button dal pannello Componenti allo stage.

**8.** Selezionare il componente Button e digitare **my\_button** nella casella di testo Nome istanza.

**9.** Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_button.label = "Seek to point2";

function clickMe(){
 my_flvPb.seekToNavCuePoint("point2");
}
my_button.addEventListener("click", clickMe);
```

**10.** Selezionare Controllo > Prova filmato per provare il codice.

Il file cuepoints.flv contiene tre cue point di navigazione: uno quasi all'inizio, uno a metà e uno alla fine del file video. Quando si fa clic sul pulsante, l'istanza FLVPlayback cerca il cue point specificato (`point2`).

Per ulteriori informazioni sui cue point, vedere “[Uso dei cue point](#)” a pagina 526. Per ulteriori informazioni sul componente FLVPlayback, vedere “[Componente FLVPlayback \(solo Flash Professional\)](#)” a pagina 517.

## Operazioni con i metadati

Per visualizzare le informazioni sui metadati nel file FLV è possibile utilizzare il metodo `onMetaData`. I metadati comprendono informazioni sul file FLV, come durata, larghezza, altezza e frequenza dei fotogrammi. Le informazioni sui metadati aggiunte al file FLV dipendono dal software utilizzato per codificare il file FLV o da quello impiegato per inserire le informazioni sui metadati.



Se il file video non dispone di informazioni sui metadati, è possibile utilizzare gli strumenti per aggiungere tali informazioni al file.

Per lavorare con `NetStream.onMetaData` è necessario disporre di Flash Video contenente i metadati. Se si codificano file FLV utilizzando Flash 8 Video Encoder, il file FLV conterrà informazioni sui metadati (per un elenco dei metadati presenti in un file FLV codificato con Flash 8 Video Encoder, vedere l'esempio seguente).



Flash Video Exporter 1.2 e versioni successive (compreso Flash 8 Video Exporter) aggiunge i metadati ai file FLV. Anche Sorenson Squeeze 4.1 e versioni successive aggiunge i metadati ai file video.

L'esempio seguente utilizza `NetStream.onMetaData` per tenere traccia delle informazioni sui metadati di un file FLV codificato con Flash 8 Video Encoder.

### Per utilizzare `NetStream.onMetaData` per visualizzare le informazioni sui metadati:

1. Creare un nuovo file FLA denominato `flvMetadata.fla`.
2. Nel pannello Libreria (Finestra > Libreria), scegliere Nuovo Video dal menu a comparsa.
3. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
4. Fare clic su OK per creare un oggetto video.
5. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza dell'oggetto video.

6. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
7. Con l'istanza video ancora selezionata, digitare **320** nella casella di testo della larghezza e **213** in quella dell'altezza.
8. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni).
9. Immettere il codice seguente nel pannello Azioni:

```
// Crea un oggetto NetConnection.
var netConn:NetConnection = new NetConnection();
// Crea una connessione in streaming locale.
netConn.connect(null);
// Crea un oggetto NetStream e definisce una funzione onStatus().
var nStream:NetStream = new NetStream(netConn);
// Associa la sorgente video NetStream all'oggetto Video.
my_video.attachVideo(nStream);
// Imposta il tempo di buffer.
nStream.setBufferTime(30);
// Inizia la riproduzione del file FLV.
nStream.play("http://www.helpexamples.com/flash/video/
 lights_short.flv");
// Traccia i metadati.
nStream.onMetaData = function(myMeta) {
 for (var i in myMeta) {
 trace(i + ":\t" + myMeta[i])
 }
};
```

10. Selezionare Controllo > Prova filmato per provare il codice.

Nel pannello Output vengono visualizzate le seguenti informazioni:

```
canSeekToEnd:true
audiocodecid:2
audiodelay:0.038
audiodatarate:96
videocodecid:4
framerate:15
videodatarate:400
height:213
width:320
duration:8.04
```

**NOTA**

Se il video non dispone dell'audio, le informazioni sui metadati relativi all'audio (come `audiodatarate`) restituiscono `undefined` perché durante la codifica non è stata aggiunta ai metadati alcuna informazione sull'audio.

Per visualizzare la maggior parte delle informazioni sui metadati è anche possibile utilizzare il formato seguente. Per esempio, il codice seguente mostra la durata di un file FLV:

```
nStream.onMetaData = function(myMeta) {
 trace("FLV duration: " + myMeta.duration + " sec.");
};
```

Questo formato non può tenere traccia delle informazioni sui metadati `cuePoint`. Per informazioni sul tracciamento dei cue point, vedere “[Tracciamento dei cue point da un file FLV](#)” a pagina 668.

## Informazioni sulla configurazione di file FLV per l'hosting su un server

Quando si lavora con file FLV, è necessario configurare il server così che possa supportare il formato di file FLV. Multipurpose Internet Mail Extensions (MIME) è una specifica di dati standardizzata che consente di inviare file non ASCII attraverso connessioni Internet. I browser Web e i client di posta elettronica sono configurati in modo da essere in grado di interpretare diversi *tipi MIME*, così che possano inviare e ricevere video, audio, immagini e testo formattato. Per caricare file FLV da un server Web, potrebbe essere necessario registrare l'estensione e il tipo MIME del file con il server Web. A questo proposito, consultare la documentazione del server Web. Il tipo MIME per i file FLV è `video/x-flv`. Informazioni complete per il tipo di file FLV sono riportate di seguito:

Tipo Mime: `video/x-flv`

Estensione del file: `.flv`

Parametri necessari: nessuno

Parametri facoltativi: nessuno

Considerazioni sulla codifica: I file FLV sono file binari; alcune applicazioni possono richiedere l'impostazione del sottotipo `application/octet-stream`.

Problemi di sicurezza: nessuno

Specifiche pubblicate: [www.macromedia.com/go/flashfileformat](http://www.macromedia.com/go/flashfileformat).

Rispetto alle versioni precedenti, Microsoft ha cambiato il modo di gestire i media in streaming nel server Web Microsoft Internet Information Services (IIS) 6.0. Le versioni precedenti di IIS non richiedono alcuna modifica per eseguire lo streaming di Flash Video. In IIS 6.0, il server Web predefinito fornito con Windows 2003 , il server richiede un tipo MIME per riconoscere che i file FLV sono media in streaming.

Quando i file SWF che eseguono lo streaming di file FLV esterni vengono posizionati su un server Microsoft Windows 2003 e visualizzati in un browser, il file SWF viene riprodotto correttamente, mentre lo streaming del video FLV non viene eseguito. Questo problema riguarda tutti i file FLV posizionati su un server Windows 2003, compresi quelli realizzati con versioni precedenti dello strumento di creazione Flash, Macromedia Flash Video Kit per Dreamweaver MX 2004. Questi file funzionano correttamente se provati su altri sistemi operativi.

Per informazioni sulla configurazione di Microsoft Windows 2003 e Microsoft IIS Server 6.0 per eseguire lo streaming di video FLV, vedere [www.macromedia.com/go/tn\\_19439](http://www.macromedia.com/go/tn_19439).

## Informazioni sull'indirizzamento di file FLV locali su Macintosh

Se ci cerca di riprodurre un file FLV locale da un'unità non di sistema su un computer Macintosh utilizzando un percorso contenente una barra relativa (/), il video non verrà riprodotto. *Le unità non di sistema* comprendono, ma non sono limitate a, CD-ROM, dischi rigidi con partizioni, unità di memorizzazione rimovibili e dispositivi di memorizzazione connessi.



Il motivo di questo fallimento riguarda una limitazione del sistema operativo e non di Flash o Flash Player.

Per riprodurre un file FLV proveniente da un'unità non di sistema su Macintosh, fare riferimento a esso con un percorso assoluto utilizzando una notazione basata sui due punti (:) al posto di una notazione basata sulla barra laterale (/). L'elenco seguente mostra le differenze tra i due tipi di notazione:

**Notazione basata sulla barra laterale** myDrive/myFolder/myFLV.flv

**Notazione basata sui due punti** (Macintosh) myDrive:myFolder:myFLV.flv

È anche possibile creare un file di proiettore per un CD-ROM che si prevede di utilizzare per la riproduzione su Macintosh. Per informazioni aggiornate sui CD-ROM Macintosh e i file FLV, vedere [www.macromedia.com/go/3121b301](http://www.macromedia.com/go/3121b301).

# Informazioni sulla creazione di animazioni di avanzamento per file multimediali

ActionScript offre diverse modalità per precaricare contenuti multimediali esterni e controllare l'avanzamento dello scaricamento di tali contenuti. Per mostrare visivamente lo stato di caricamento o la quantità di contenuto che è stato caricato è possibile creare barre o animazioni di avanzamento.

Per precaricare file SWF e JPEG, utilizzare la classe MovieClipLoader, che fornisce un meccanismo listener di eventi per controllare l'avanzamento dello scaricamento. Per ulteriori informazioni, vedere "[Percaricamento di file SWF e JPEG](#)" a pagina 427.

Per controllare lo stato dello scaricamento di file MP3, utilizzare i metodi Sound.getBytesLoaded() e Sound.getBytesTotal(); per controllare lo stato dello scaricamento di file FLV, utilizzare le proprietà NetStream.bytesLoaded e NetStream.bytesTotal. Per ulteriori informazioni, vedere "[Percaricamento di file MP3](#)" a pagina 420.

Per informazioni sulla creazione di barre di avanzamento per caricare file multimediali, consultare i seguenti argomenti:

- ["Creazione di un'animazione di avanzamento per caricare file SWF e di immagine"](#)  
a pagina 682
- ["Creazione di una barra di avanzamento per caricare file MP3 con ActionScript"](#)  
a pagina 684
- ["Creazione di una barra di avanzamento per caricare file FLV con ActionScript"](#)  
a pagina 686

Per creare un'animazione per una barra di avanzamento è possibile trovare un file sorgente di esempio che utilizza animazioni con script. Individuare tweenProgressBar.fla nella cartella Samples sul disco rigido:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar*.

## Creazione di un'animazione di avanzamento per caricare file SWF e di immagine

Quando si caricano file SWF o di immagine di grandi dimensioni in un'applicazione, si può desiderare di creare un'animazione che mostri lo stato del caricamento. È possibile creare una barra di avanzamento che mostri i progressi nel caricamento dell'animazione. È anche possibile creare un'animazione che cambia mentre il file viene caricato. Per ulteriori informazioni sul caricamento dei file SWF e dei file di immagine, vedere “[Caricamento di file SWF e file di immagine esterni](#)” a pagina 647.

L'esempio seguente mostra come utilizzare la classe MovieClipLoader e l'API di disegno per visualizzare l'avanzamento del caricamento di un file di immagine.

### Per creare una barra di avanzamento per caricare file di immagine o SWF:

1. Creare un nuovo documento Flash denominato **loadImage.fla**.
2. Selezionare Elabora > Documento e digitare **700** nella casella di testo della larghezza e **500** in quella dell'altezza per modificare le dimensioni del documento.
3. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, digitare il codice seguente:

```
//crea clip per ospitare il contenuto
this.createEmptyMovieClip("progressBar_mc", 0);
progressBar_mc.createEmptyMovieClip("bar_mc", 1);
progressBar_mc.createEmptyMovieClip("stroke_mc", 2);
//utilizza i metodi di disegno per creare una barra di avanzamento
with (progressBar_mc.stroke_mc) {
 lineStyle(0, 0x000000);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 10);
 lineTo(0, 10);
 lineTo(0, 0);
}
with (progressBar_mc.bar_mc) {
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 10);
 lineTo(0, 10);
 lineTo(0, 0);
 endFill();
 _xscale = 0;
}
progressBar_mc._x = 2;
progressBar_mc._y = 2;
// avanzamento del caricamento
```

```

var mcListener:Object = new Object();
mcListener.onLoadStart = function(target_mc:MovieClip) {
 progressBar_mc.bar_mc._xscale = 0;
};
mcListener.onLoadProgress = function(target_mc:MovieClip,
 bytesLoaded:Number, bytesTotal:Number) {
 progressBar_mc.bar_mc._xscale = Math.round(bytesLoaded/
 bytesTotal*100);
};
mcListener.onLoadComplete = function(target_mc:MovieClip) {
 progressBar_mc.removeMovieClip();
};
mcListener.onLoadInit = function(target_mc:MovieClip) {
 target_mc._height = 500;
 target_mc._width = 700;
};
// Crea un clip per ospitare l'immagine.
this.createEmptyMovieClip("image_mc", 100);
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mcListener);
/* Carica l'immagine nel clip.
You can change the following URL to a SWF or another image file. */
image_mcl.loadClip("http://www.helpexamples.com/flash/images/gallery1/
 images/pic3.jpg", image_mc);

```

- 4.** Selezionare Controllo > Prova filmato per visualizzare il caricamento dell'immagine e osservare la barra di avanzamento.

**NOTA**

Se si prova questo codice una seconda volta, l'immagine verrà memorizzata nella cache e la barra di avanzamento si completerà immediatamente. Per provare più volte, utilizzare immagini diverse e cariarle da una sorgente esterna. Una sorgente locale può causare problemi durante la verifica delle applicazioni perché il contenuto viene caricato troppo velocemente.

Per creare un'animazione per una barra di avanzamento è possibile trovare un file sorgente di esempio che utilizza animazioni con script. Individuare tweenProgress.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar*.

È anche possibile trovare esempi di applicazioni di gallerie fotografiche. Questi file forniscono esempi sull'utilizzo di ActionScript per controllare clip filmato in modo dinamico mentre si caricano file di immagine in un file SWF. È possibile trovare i file sorgente di esempio, *gallery\_tree.fla* e *gallery\_tween.fla*, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Galleries*.

## Creazione di una barra di avanzamento per caricare file MP3 con ActionScript

Nell'esempio seguente vengono caricate più canzoni in un file SWF. Nella barra di avanzamento, creata mediante l'API di disegno, viene visualizzato l'avanzamento del caricamento. Quando la musica inizia e viene completato il caricamento, le relative informazioni vengono visualizzate nel pannello Output. Per informazioni sul caricamento di file MP3, vedere “[Caricamento di un file MP3](#)” a pagina 654.

### Per creare una barra di avanzamento per caricare file MP3:

1. Creare un nuovo documento Flash denominato **loadSound.fla**.
2. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
 this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
 pb_width, 22);

pb._x = 100;
pb._y = 100;

with (pb.bar_mc) {
 beginFill(0x00FF00);
 moveTo(0, 0);
 lineTo(pb_width, 0);
 lineTo(pb_width, pb_height);
 lineTo(0, pb_height);
 lineTo(0, 0);
 endFill();
```

```

 _xscale = 0;
 }
 with (pb.vBar_mc) {
 lineStyle(1, 0x000000);
 moveTo(0, 0);
 lineTo(0, pb_height);
 }
 with (pb.stroke_mc) {
 lineStyle(3, 0x000000);
 moveTo(0, 0);
 lineTo(pb_width, 0);
 lineTo(pb_width, pb_height);
 lineTo(0, pb_height);
 lineTo(0, 0);
 }

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
 if (success) {
 trace("sound loaded");
 }
};
my_sound.onSoundComplete = function() {
 clearInterval(my_interval);
 trace("Cleared interval");
}
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3",
 true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
 var pos:Number = Math.round(the_sound.position / the_sound.duration * 100);
 pb.bar_mc._xscale = pos;
 pb.vBar_mc._x = pb.bar_mc._width;
 pb.pos_txt.text = pos + "%";
}

```

3. Selezionare Controllo > Prova filmato per caricare il file MP3 e osservare la barra di avanzamento.

**NOTA**

Se si prova questo codice una seconda volta, l'immagine verrà memorizzata nella cache e la barra di avanzamento si completerà immediatamente. Per provare più volte, utilizzare immagini diverse e caricarle da una sorgente esterna. Una sorgente locale può causare problemi durante la verifica delle applicazioni perché il contenuto viene caricato troppo velocemente.

Per ulteriori informazioni sull'utilizzo dell'audio, consultare le voci della classe Sound, Sound nella *Guida di riferimento di ActionScript 2.0*.

Per creare un'animazione per una barra di avanzamento è possibile trovare un file sorgente di esempio che utilizza animazioni con script. Individuare tweenProgress.fla nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar*.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio che carica file MP3, jukebox.fla. Questo esempio illustra come creare un jukebox utilizzando i tipi di dati, i principi di codificazione generali e diversi componenti:

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox*.

## Creazione di una barra di avanzamento per caricare file FLV con ActionScript

Per visualizzare lo stato di caricamento di un file FLV è possibile creare una barra di avanzamento. Per informazioni sul caricamento di file FLV in un file SWF, vedere “[Precarica di file FLV](#)” a pagina 665. Per ulteriori informazioni sui file FLV e Flash, vedere “[Informazioni sull'uso dei file video FLV](#)” a pagina 659.

L'esempio seguente utilizza l'API di disegno per creare una barra di avanzamento. L'esempio utilizza anche le proprietà bytesLoaded e bytesTotal che mostrano lo stato del caricamento del file video1.flv nell'istanza dell'oggetto video denominata my\_video. Viene creato dinamicamente un campo di testo, denominato loaded\_txt, per visualizzare le informazioni sull'avanzamento del caricamento.

### Per creare una barra di avanzamento che mostri lo stato del caricamento:

1. Creare un nuovo file FLA denominato flvProgress.fla.
2. Nel pannello Libreria (Finestra > Libreria), scegliere Nuovo Video dal menu a comparsa.
3. Nella finestra di dialogo Proprietà video assegnare un nome al simbolo del video e selezionare Video (controllato da ActionScript).
4. Fare clic su OK per creare un oggetto video.

5. Trascinare l'oggetto video dal pannello Libreria allo stage per creare un'istanza dell'oggetto video.
6. Con l'oggetto video selezionato sullo stage, digitare **my\_video** nella casella di testo Nome istanza della finestra di ispezione Proprietà (Finestra > Proprietà > Proprietà).
7. Con l'istanza video selezionata, digitare **320** nella casella di testo della larghezza e **213** in quella dell'altezza.
8. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, immettere il codice seguente:

```

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
 typing_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
 160, 22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
 progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
 beginFill(0xFF0000);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 10);
 lineTo(0, 10);
 lineTo(0, 0);
 endFill();
 _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
 progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
 lineStyle(0, 0x000000);
 moveTo(0, 0);
 lineTo(100, 0);
 lineTo(100, 10);
 lineTo(0, 10);
 lineTo(0, 0);
}

```

```

var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
 stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
 var pctLoaded:Number = Math.round(my_ns.bytesLoaded /
 my_ns.bytesTotal * 100);
 loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
 Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
 "%)";
 progressBar_mc.bar_mc._xscale = pctLoaded;
 if (pctLoaded>=100) {
 clearInterval(loaded_interval);
 }
}

```

**9.** Selezionare Controllo > Prova filmato per provare il codice.

Il video viene caricato e i valori di animazione della barra e di modifica del testo comunicano l'avanzamento dello scaricamento. Se questi elementi coprono il video, spostare l'oggetto video sullo stage. È possibile personalizzare il colore della barra di avanzamento modificando `beginFill` e `lineStyle` nel frammento di codice precedente.

**NOTA**

Se la barra di avanzamento viene caricata subito, il video è stato memorizzato nella cache del disco rigido (o perché l'esempio è già stato provato o perché è stato caricato in una procedura diversa). Se si verifica questa situazione, inserire un file FLV nel server e caricarlo.

Per creare un'animazione per una barra di avanzamento è possibile trovare un file sorgente di esempio che utilizza animazioni con script. Individuare `tweenProgress.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/Tween ProgressBar*.

# Operazioni con i dati esterni

In Macromedia Flash Basic 8 e Macromedia Flash Professional 8, è possibile utilizzare ActionScript per caricare dati da origini esterne in un file SWF. Si possono anche inviare dati che potrebbero essere forniti dall'utente o dal server, da un file SWF ad un server di applicazioni (come Macromedia ColdFusion o Macromedia JRun) o ad un altro tipo di script sul lato server come PHP o Perl. Macromedia Flash Player può inviare e caricare dati tramite HTTP o HTTPS oppure da un file di testo locale. È possibile anche creare connessioni socket TCP/IP persistenti per applicazioni che richiedono bassa latenza, ad esempio applicazioni chat o servizi per quotazioni azionarie. Una delle nuove funzioni in FlashPlayer 8 è la capacità di caricare file dal computer dell'utente ad un server e scaricare file dal server al computer dell'utente.

I dati caricati o inviati da un file SWF possono essere formattati come XML (Extensible Markup Language) o come coppie nome-valore.

Flash Player può inoltre inviare e ricevere dati dal proprio ambiente host, ad esempio un browser, o da un'altra istanza di Flash Player sullo stesso computer o nella stessa pagina Web.

Per impostazione predefinita, un file SWF può accedere solo a dati presenti nello stesso dominio (ad esempio [www.macromedia.com](http://www.macromedia.com)). Per ulteriori informazioni, consultare “[Informazioni su domini, sicurezza tra domini e file SWF](#)” a pagina 753.

Per ulteriori informazioni sulle operazioni con dati esterni, consultare i seguenti argomenti:

|                                                                               |     |
|-------------------------------------------------------------------------------|-----|
| <a href="#">Invio e caricamento di variabili</a> .....                        | 690 |
| <a href="#">Uso di HTTP per la connessione a script sul lato server</a> ..... | 695 |
| <a href="#">Informazioni sul caricamento e scaricamento dei file</a> .....    | 700 |
| <a href="#">Informazioni su XML</a> .....                                     | 709 |
| <a href="#">Invio di messaggi verso e da Flash Player</a> .....               | 719 |
| <a href="#">Informazioni sull'API External</a> .....                          | 723 |

# Invio e caricamento di variabili

Un file SWF può essere paragonato a una finestra per l'acquisizione e la visualizzazione di informazioni, come una pagina HTML. I file SWF, però, rimangono caricati nel browser e vengono aggiornati in continuazione con nuove informazioni senza necessità di ricaricare tutta la pagina. Tramite le funzioni e i metodi di ActionScript è possibile inviare e ricevere informazioni da script sul lato server e ricevere informazioni da file di testo e XML.

Tramite script sul lato server è inoltre possibile richiedere informazioni specifiche da un database e inoltrarle a un file SWF. Gli script sul lato server possono essere scritti in linguaggi diversi. Quelli più utilizzati sono CFML, Perl, ASP (Microsoft Active Server Pages) e PHP. Memorizzando informazioni in un database e recuperandole, è possibile creare contenuto dinamico e personalizzato per il file SWF, ad esempio un forum, profili personali per gli utenti o un carrello in cui registrare gli acquisti di un utente.

Per inviare e ottenere le informazioni da un file SWF sono disponibili diversi metodi e funzioni ActionScript. Ogni funzione o metodo utilizza un particolare protocollo per il trasferimento delle informazioni e richiede una formattazione specifica delle informazioni.

- Le funzioni e i metodi MovieClip che utilizzano il protocollo HTTP o HTTPS per inviare le informazioni con codifica URL sono `getURL()`, `loadVariables()`, `loadVariablesNum()`, `loadMovie()` e `loadMovieNum()`.
- I metodi LoadVars che utilizzano il protocollo HTTP o HTTPS per inviare e caricare informazioni in formato con codifica URL sono `load()`, `send()` e `sendAndLoad()`.
- I metodi che utilizzano il protocollo HTTP o HTTPS per inviare e caricare informazioni in formato XML sono `XML.send()`, `XML.load()` e `XML.sendAndLoad()`.
- I metodi che creano e utilizzano una connessione socket TCP/IP per inviare e caricare informazioni in formato XML sono `XMLSocket.connect()` e `XMLSocket.send()`.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Controllo dei dati caricati](#)” a pagina 691
- “[Creazione di una barra di avanzamento per visualizzare l'avanzamento del caricamento dei dati](#)” a pagina 692

## Controllo dei dati caricati

Ogni funzione o metodo che carica dati in un file SWF, ad eccezione di `XMLSocket.send()`, è detto *asincrono*, perché i risultati dell'azione vengono restituiti in un momento indeterminato.

Prima di utilizzare i dati caricati in un file SWF, è necessario verificare che siano stati caricati. Non è possibile, ad esempio, caricare variabili e gestirne i valori nello stesso script, perché i dati da gestire non sono presenti nel file prima che siano stati caricati. Nello script seguente, non è possibile utilizzare la variabile `lastSiteVisited` fino a quando non è stata caricata dal file `myData.txt`. Nel file `myData.txt` sarà presente un testo analogo a quello dell'esempio seguente:

```
lastSiteVisited=www.macromedia.com
```

Se invece si utilizza il codice seguente, non sarà possibile tenere traccia dei dati che vengono caricati:

```
loadVariables("myData.txt", 0);
trace(lastSiteVisited); // undefined
```

Ogni funzione o metodo dispone di una tecnica particolare per controllare i dati caricati. Se si utilizza `loadVariables` function o `loadMovie` function è possibile caricare le informazioni in un target rappresentato da un clip filmato e utilizzare il gestore `onData` per eseguire uno script. Se si caricano i dati tramite `loadVariables` function, il gestore `onData` viene eseguito dopo il caricamento dell'ultima variabile. Se invece i dati vengono caricati con `loadMovie` function, il gestore `onData` viene eseguito ogni volta che ha luogo lo streaming di un frammento del file SWF in Flash Player.

Il seguente codice ActionScript, ad esempio, carica le variabili dal file `myData.txt` nel clip filmato `loadTarget_mc`. Un gestore `onData()` assegnato all'istanza `loadTarget_mc` utilizza la variabile `lastSiteVisited`, caricata dal file `myData.txt`. Le seguenti azioni di analisi vengono eseguite solo dopo il caricamento di tutte le variabili, compresa `lastSiteVisited`.

```
this.createEmptyMovieClip("loadTarget_mc", this.getNextHighestDepth());
this.loadTarget_mc.onData = function() {
 trace("Dati caricati");
 trace(this.lastSiteVisited);
};
loadVariables("myData.txt", this.loadTarget_mc);
```

Se si utilizzano i metodi `XML.load()`, `XML.sendAndLoad()` e `XMLSocket.connect()`, occorre definire un gestore che elabori i dati quando vengono ricevuti. Questo gestore è una proprietà di un oggetto `XML` o `XMLSocket` al quale viene assegnata una funzione personalizzata. I gestori vengono chiamati automaticamente alla ricezione delle informazioni. Per l'oggetto `XML`, utilizzare `XML.onLoad()` o `XML.onData()`. Per l'oggetto `XMLSocket` utilizzare `XMLSocket.onConnect()`.

Per ulteriori informazioni, vedere “Uso della classe XML” a pagina 711 e “Uso della classe XMLSocket” a pagina 717. Per ulteriori informazioni sull'uso di LoadVars per inviare e caricare dati elaborabili dopo la loro ricezione, consultare “Uso della classe LoadVars” a pagina 696.

## Creazione di una barra di avanzamento per visualizzare l'avanzamento del caricamento dei dati

L'esercizio seguente crea in modo dinamico un semplice precaricatore con l'API (Application Programming Interface) di disegno e visualizza l'avanzamento del caricamento di un documento XML.

SUGGERIMENTO

Se si carica il file XML remoto troppo rapidamente, per vedere l'effetto di precaricamento, cercare di caricare un file XML più grande in Internet, quindi caricare il file.

### Creazione di una barra di avanzamento tramite l'API di disegno

1. Creare un nuovo documento Flash e salvarlo come **drawapi.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var barWidth:Number = 200;
var barHeight:Number = 6;

this.createEmptyMovieClip("pBar_mc", 9999);
var bar:MovieClip = pBar_mc.createEmptyMovieClip("bar_mc", 10);
bar.beginFill(0xFF0000, 100);
bar.moveTo(0, 0);
bar.lineTo(barWidth, 0);
bar.lineTo(barWidth, barHeight);
bar.lineTo(0, barHeight);
bar.lineTo(0, 0);
bar.endFill();
bar._xscale = 0;

var stroke:MovieClip = pBar_mc.createEmptyMovieClip("stroke_mc", 20);
stroke.lineStyle(0, 0x000000);
stroke.moveTo(0, 0);
stroke.lineTo(barWidth, 0);
stroke.lineTo(barWidth, barHeight);
stroke.lineTo(0, barHeight);
stroke.lineTo(0, 0);
```

```

pBar_mc.createTextField("label_txt", 30, 0, barHeight, 100, 21);
pBar_mc.label_txt.autoSize = "left";
pBar_mc.label_txt.selectable = false;

pBar_mc._x = (Stage.width - pBar_mc._width) / 2;
pBar_mc._y = (Stage.height - pBar_mc._height) / 2;

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
 pBar_mc.onEnterFrame = undefined;
 if (success) {
 trace("Caricamento XML completato");
 } else {
 trace("Impossibile caricare XML");
 }
};
my_xml.load("http://www.helpexamples.com/flash/xml/ds.xml");

pBar_mc.onEnterFrame = function() {
 var pctLoaded:Number = Math.floor(my_xml.getBytesLoaded() /
 my_xml.getBytesTotal() * 100);
 if (!isNaN(pctLoaded)) {
 pBar_mc.bar_mc._xscale = pctLoaded;
 pBar_mc.label_txt.text = pctLoaded + "% loaded";
 if (pctLoaded >= 100)
 pBar_mc.onEnterFrame = undefined;
 }
}
};


```

Il codice precedente viene suddiviso in sette sezioni. La prima sezione definisce la larghezza e l'altezza della barra di avanzamento, se disegnata sullo stage. La barra verrà centrata sullo stage in una sezione successiva. La sezione successiva di codice crea due clip filmato vuoti, pBar\_mc e bar\_mc. Il clip filmato bar\_mc viene nidificato all'interno di pBar\_mc e disegna un rettangolo rosso nello stage. L'istanza bar\_mc modifica la relativa proprietà `x_scale` quando si carica il file XML esterno dal sito Web remoto.

In seguito, viene nidificato il secondo clip filmato nel clip pBar\_mc, stroke\_mc. Il clip filmato stroke\_mc traccia la struttura nello stage che corrisponde alle dimensioni specificate dalle variabili barHeight e barWidth definite nella prima sezione. La quarta sezione del codice crea nel clip filmato pBar\_mc un campo di testo utilizzato per visualizzare la percentuale già caricata del file XML, simile all'etichetta sul componente ProgressBar. Quindi, il clip filmato pBar\_mc, che include le istanze nidificate bar\_mc, stroke\_mc e label\_txt, viene centrato nello stage.

La sesta sezione del codice definisce una nuova istanza di oggetto XML utilizzata per caricare un file XML esterno. Viene definito un gestore di eventi `onLoad` che traccia un messaggio sul pannello Output. Tale gestore elimina anche il gestore di eventi `onEnterFrame`, definito nella sezione successiva, per il clip filmato `pBar_mc`. La sezione finale del codice definisce un gestore di eventi `onEnterFrame` per il clip filmato `pBar_mc`. Tale gestore monitora la quantità caricata del file XML esterna e modifica la proprietà `_xscale` per il clip filmato `bar_mc`. Prima il gestore di eventi `onEnterFrame` calcola la percentuale del file per terminare il caricamento. Se la percentuale del file caricato è un numero valido, viene impostata la proprietà `_xscale` per `bar_mc` e il campo di testo in `pBar_mc` visualizza la percentuale del file caricata. Se il caricamento del file è terminato (ovvero la percentuale raggiunge il 100%), il gestore di eventi `onEnterFrame` viene eliminato così da non monitorare più l'avanzamento del download.

**3.** Selezionare Controllo > Prova filmato per provare il documento Flash.

Quando il file XML esterno viene caricato, il clip filmato nidificato `bar_mc` si ridimensiona per visualizzare lo stato di avanzamento dello scaricamento del file XML. Una volta completato il caricamento del file, il gestore di eventi `onEnterFrame` viene eliminato così da non continuare a calcolare lo stato di avanzamento dello scaricamento. A seconda della velocità con cui lo scaricamento viene completato, dovrebbe essere possibile visualizzare il lento ingrandimento della barra finché `bar_mc` non raggiunge la stessa larghezza del clip filmato `stroke_mc`. Se lo scaricamento avviene troppo velocemente, la barra di avanzamento potrebbe passare da 0% a 100% troppo rapidamente, rendendo difficile la visualizzazione; in tal caso potrebbe essere necessario cercare di scaricare un file XML più grande.

# Uso di HTTP per la connessione a script sul lato server

Le funzioni `loadVariables` function, `loadVariablesNum` function, `getURL` function, `loadMovie` function, `loadMovieNum` function e `loadVariables` (`MovieClip.loadVariables` method), `loadMovie` (`MovieClip.loadMovie` method) nonché i metodi `getURL` (`MovieClip.getURL` method) possono comunicare con gli script sul lato server mediante protocolli HTTP o HTTPS. Tali funzioni e metodi inviano tutte le variabili dalla linea temporale alla funzione a cui sono collegati. Quando vengono utilizzati come metodi dell'oggetto `MovieClip`, `loadVariables()`, `getURL()` e `loadMovie()`, inviano tutte le variabili del clip filmato specificato. Ogni funzione o metodo gestisce la risposta come indicato di seguito:

- La funzione `getURL()` restituisce informazioni a una finestra del browser e non a Flash Player.
- I metodi `loadVariables()` caricano le variabili in Flash Player in una linea temporale o un livello specifico.
- Il metodo `loadMovie()` carica un file SWF in Flash Player in un livello o un clip filmato specifico.

Con `loadVariables()`, `getURL()` o `loadMovie()` è possibile specificare diversi parametri:

- *URL* corrisponde al file in cui si trovano le variabili remote.
- *Percorso* è il livello o il target nel file SWF che riceve le variabili. La funzione `getURL()` non accetta questo parametro.

Per ulteriori informazioni su livelli e target, consultare [Capitolo 1, “Informazioni su livelli e linee temporali multipli”](#) in *Uso di Flash*.

- *Variabili* imposta il metodo HTTP di invio delle variabili, ovvero `GET` (le variabili vengono aggiunte al termine dell'URL) o `POST` (le variabili vengono inviate in un'intestazione HTTP separata). Se questo parametro viene omesso, viene utilizzato il valore predefinito `GET`, ma non vengono inviate variabili.

Se, ad esempio, si desidera tenere traccia dei migliori punteggi di un gioco, è possibile memorizzare i punteggi su un server e utilizzare `loadVariables()` per caricarli nel file SWF quando un giocatore termina una partita. La funzione potrebbe avere un aspetto analogo al seguente:

```
this.createEmptyMovieClip("highscore_mc", 10);
loadVariables("http://www.helpexamples.com/flash/highscore.php",
 highscore_mc, "GET");
```

In questo esempio vengono caricate variabili dallo script ColdFusion `high_score.cfm` nell'istanza del clip filmato `scoreClip` con l'ausilio del metodo HTTP GET.

Le variabili caricate con la funzione `loadVariables()` devono avere il formato MIME standard *application/x-www-form-urlencoded* (un formato standard utilizzato negli script CFM e CGI). Il file specificato nel parametro dell'*URL* di `loadVariables()` deve riportare le coppie variabile-valore in questo formato in modo che sia leggibile da parte di Flash. Questo file può specificare qualsiasi numero di variabile; le coppie variabile-valore devono essere separate da una e commerciale (`&`) e tutte le parole interne al valore devono essere separate dal segno più (`+`). L'espressione seguente definisce diverse variabili:

```
highScore1=54000&playerName1=RGoulet&highScore2=53455&playerName2=
WNewton&highScore3=42885&playerName3=TJones
```

**NOTA**

Alcuni caratteri, ad esempio il segno più (`+`) o la e commerciale (`&`), potrebbero necessitare della codifica URL. Per ulteriori informazioni, consultare il sito web [www.macromedia.com/go/tn\\_14143](http://www.macromedia.com/go/tn_14143).

Per ulteriori informazioni, consultare i seguenti argomenti: “[Uso della classe LoadVars](#)” [a pagina 696](#). Vedere anche la voce `loadVariables` function, `getURL` function, `loadMovie` function e `LoadVars` nella *Guida di riferimento di ActionScript 2.0*.

## Uso della classe LoadVars

Se si esegue la pubblicazione per Flash Player 6 o versioni successive e si desidera ottenere una maggiore flessibilità rispetto a quanto assicurato da `loadVariables()`, è possibile fare ricorso alla classe `LoadVars`, anziché trasferire variabili tra un file SWF e un server.

La classe `LoadVars` è stata introdotta con Flash Player 6 per garantire un'interfaccia più semplice e orientata agli oggetti per le attività comuni di scambio di dati CGI con un server Web. I vantaggi derivanti dall'uso di `LoadVars` comprendono quanto segue:

- Non è necessario creare un clip filmato contenitore per i dati o inserire nei clip filmato esistenti variabili specifiche della comunicazione client/server.
- L'interfaccia della classe è simile all'oggetto XML e garantisce quindi uniformità all'interno del codice ActionScript. Utilizza infatti i metodi `load()`, `send()` e `sendAndLoad()` per avviare la comunicazione con un server. La differenza principale tra le classi `LoadVars` e XML sta nel fatto che i dati di `LoadVars` sono rappresentati da una proprietà dell'oggetto `LoadVars` anziché da una struttura ad albero DOM (Document Object Model) XML memorizzata nell'oggetto XML.
- L'interfaccia della classe è più semplice rispetto all'interfaccia di `loadVariables`; i metodi sono denominati semplicemente `load`, `send`, `sendAndLoad`.

- È possibile ottenere informazioni aggiuntive sulla comunicazione tramite i metodi `getBytesLoaded` e `getBytesTotal`.
- È possibile ottenere informazioni sullo stato di scaricamento dei dati, sebbene non sia possibile accedere ai dati prima del loro caricamento completo.
- L'interfaccia di callback viene implementata tramite i metodi ActionScript (`onLoad`), anziché tramite l'approccio obsoleto con `onClipEvent`, necessario per `loadVariables`.
- Sono disponibili notifiche per gli errori.
- È possibile aggiungere intestazioni di richiesta HTTP personalizzate.

È necessario creare un oggetto `LoadVars` per chiamarne i metodi. Questo oggetto è un contenitore per i dati caricati.

Nella procedura seguente viene illustrato come utilizzare ColdFusion e la classe `LoadVars` per inviare un messaggio e-mail da un file SWF.



Per questo esempio, sul server Web deve essere installato ColdFusion.

#### **Per caricare i dati con l'oggetto `LoadVars`:**

1. Creare un file CFM in Macromedia Dreamweaver o nell'editor di testo preferito.  
Aggiungere al file il testo seguente:  

```
<cfif StructKeyExists(Form, "emailTo")>
<cfmail to="#Form.emailTo#" from="#Form.emailFrom#"
 subject="#Form.emailSubject#">#Form.emailBody#</cfmail>
&result=true
<cfelse>
&result=false
</cfif>
```
2. Salvare il file come `email.cfm` e caricarlo sul sito Web.
3. Creare un nuovo documento in Flash.
4. Creare quattro campi di testo di input sullo stage e assegnare i seguenti nomi di istanza: `emailFrom_txt`, `emailTo_txt`, `emailSubject_txt` e `emailBody_txt`.
5. Creare un campo di testo dinamico sullo stage con il nome di istanza `debug_txt`.
6. Creare un simbolo di pulsante, trascinare un'istanza sullo stage e assegnarvi il nome di istanza `submit_btn`.
7. Selezionare il fotogramma 1 nella linea temporale e aprire il pannello Azioni (Finestra > Azioni), se non è già visualizzato.

**8.** Immettere il codice seguente nel pannello Azioni:

```
this.submit_btn.onRelease = function(){
 var emailResponse:LoadVars = new LoadVars();
 emailResponse.onLoad = function(success:Boolean) {
 if (success) {
 debug_txt.text = this.result;
 } else {
 debug_txt.text = "errore scaricamento contenuto";
 }
 };
 var email:LoadVars = new LoadVars();
 email.emailFrom = emailFrom_txt.text;
 email.emailTo = emailTo_txt.text;
 email.emailSubject = emailSubject_txt.text;
 email.emailBody = emailBody_txt.text;
 email.sendAndLoad("http://www.yoursite.com/email.cfm", emailResponse,
 "POST");
};
```

Questo codice ActionScript crea una nuova istanza dell'oggetto LoadVars, copia i valori dai campi di testo nell'istanza e invia i dati al server. Il file CFM invia il messaggio e-mail e restituisce una variabile (true o false) al file SWF chiamato result che viene visualizzata nel campo di testo debug\_txt.



Ricordare di modificare l'URL www.yoursite.com inserendo il nome del proprio dominio.

- 9.** Salvare il documento come **sendEmail.fla** e pubblicarlo scegliendo File > Pubblica.
- 10.** Caricare sendEmail.swf nella stessa directory di email.cfm (il file di ColdFusion salvato e caricato al punto 2).
- 11.** Visualizzare e provare il file SWF in un browser.

Per ulteriori informazioni, vedere la voce LoadVars nella *Guida di riferimento di ActionScript 2.0*.

Flash Player 8 ha introdotto il gestore di eventi `onHTTPStatus` per la classe LoadVars, la classe XML e la classe MovieClipLoader per permettere agli utenti di accedere al codice di stato da una richiesta HTPP. Questa funzionalità consente agli sviluppatori di determinare il *perché* una particolare operazione di caricamento potrebbe non aver dato esito positivo piuttosto che determinare solo che tale operazione ha già dato esito negativo.

L'esempio seguente mostra come poter utilizzare il gestore eventi `onHTTPStatus` della classe LoadVars per controllare se un file di testo è stato scaricato dal server e quale codice di stato è stato restituito dalla richiesta HTTP.

### Per verificare lo stato HTPP con l'oggetto LoadVars:

1. Creare un nuovo documento Flash e salvarlo come **loadvars.fla**.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
this.createTextField("params_txt", 10, 10, 10, 100, 21);
params_txt.autoSize = "left";

var my_lv:LoadVars = new LoadVars();
my_lv.onHTTPStatus = function(httpStatus:Number) {
 trace("HTTP status is: " + httpStatus);
};
my_lv.onLoad = function(success:Boolean) {
 if (success) {
 trace("text file successfully loaded");
 params_txt.text = my_lv.dayNames;
 } else {
 params_txt.text = "unable to load text file";
 }
};
my_lv.load("http://www.helpexamples.com/flash/404.txt");
/* output:
 Error opening URL "http://www.helpexamples.com/flash/404.txt"
 HTTP status is: 404
*/
```

Il codice precedente crea un nuovo campo di testo nello stage e permette il ridimensionamento automatico del campo di testo. Successivamente, si crea un oggetto LoadVars e due gestori di eventi: onHTTPStatus e onLoad. Il gestore di eventi onHTTPStatus è nuovo per Flash Player 8 e viene chiamato quando un'operazione LoadVars.load() o LoadVars.sendAndLoad() è terminata. Il valore passato alla funzione del gestore di eventi onHTTPStatus (httpStatus nel codice precedente) contiene la definizione del codice di stato HTPP per l'operazione di caricamento corrente. Se il file è riuscito a caricare il file di testo, il valore di httpStatus viene impostato su 200 (codice di stato HTPP "OK"). Se il file non esiste nel server, il valore di httpStatus viene impostato su 404 (codice di stato HTPP "Non trovato"). Il secondo gestore di eventi, LoadVars.onLoad(), viene chiamato al termine del caricamento del file. Se il file viene caricato, il valore del parametro success è impostato su true; in caso contrario il parametro success è impostato su false. Da ultimo, viene caricato il file esterno tramite il metodo LoadVars.load().

3. Selezionare Controllo > Prova filmato per provare il documento Flash.

Flash visualizza un messaggio di errore nel pannello Output che afferma di non essere stato in grado di caricare l'immagine perché non esiste sul server. Il gestore di eventi onHTTPStatus traccia il codice di stato 404 poiché non è stato possibile trovare il file sul server e il gestore di eventi onLoad imposta la proprietà testo del campo di testo params\_txt su "impossibile caricare il file di testo".

**ATTENZIONE**

Se un server web non restituisce il codice di stato a Flash Player, al gestore di eventi onHTTPStatus viene restituito il numero 0.

## Informazioni sul caricamento e scaricamento dei file

La classe FileReference consente di aggiungere la possibilità di caricare e scaricare file tra client e server. Gli utenti possono caricare o scaricare i file sui propri computer e sul server. Una finestra di dialogo, come Apri nel sistema operativo Windows, richiede di selezionare un file da caricare o una posizione per scaricarlo.

Ogni oggetto FileReference che si crea con ActionScript fa riferimento ad un singolo file nel disco rigido dell'utente. L'oggetto ha proprietà che contengono informazioni sulla dimensione, tipo, nome, data di creazione e modifica del file. Macintosh presenta anche la proprietà relativa al tipo di generatore del file.

È possibile creare un'istanza del FileReference in due modi. Si può utilizzare il nuovo operatore seguente:

```
import flash.net.FileReference;
var myFileReference:FileReference = new FileReference();
```

In alternativa, è possibile chiamare il metodo `FileReferenceList.browse()` che apre una finestra di dialogo nel sistema dell'utente che richiede di selezionare un file da scaricare, quindi crea un array per gli oggetti `FileReferece` qualora l'utente riesca a selezionare almeno un file. Ogni oggetto `FileReference` rappresenta un file selezionato dall'utente nella finestra di dialogo. Un oggetto `FileReference` non contiene alcun dato delle proprietà `FileReference` (come `name`, `size` o `modificationDate`) finché non si chiama il metodo `FileReference.browse()` o `FileReferenceList.browse()` e l'utente seleziona un file dal selettore o finché non si utilizza il metodo `FileReference.download()` per scegliere un file dal selettor.

**NOTA**

`FileReference.browse()` consente all'utente di selezionare un solo file.  
`FileReferenceList.browse()` consente all'utente di selezionare più file.

Dopo l'esito positivo di una chiamata al metodo `browse()`, chiamare `FileReference.upload()` per caricare un file per volta.

È possibile anche aggiungere la fuzionalità di scaricamento all'applicazione Flash. Il metodo `FileReference.download()` richiede agli utenti finali di individuare una posizione nei relativi dischi rigidi per salvare il file proveniente da un server. Inoltre, questo metodo inizializza lo scaricamento da un URL remoto. Quando si utilizza il metodo `download()`, è accessibile solo la proprietà `FileReference.name` quando si invia l'evento `onSelect`. Il resto delle proprietà non è accessibile finché non si invia l'evento `onComplete`.

**NOTA**

Quando viene visualizzata una finestra di dialogo sul computer dell'utente finale, la posizione predefinita che appare nella finestra di dialogo corrisponde all'ultima cartella visualizzata (se si può individuare la posizione) o al desktop (se non si può individuare la posizione). Le API `FileReference` e `FileReferenceList` non consentono di impostare un percorso predefinito per i file.

Per informazioni sulla funzionalità e la sicurezza dell'API `FileReference`, consultare “[Informazioni sulla funzionalità e sicurezza dell'API FileReference](#)” a pagina 702. Per informazioni sull'applicazione dell'API `FileReference`, consultare “[Aggiunta della funzionalità di caricamento file ad un'applicazione](#)” a pagina 703. È possibile trovare i file di origine di esempio, `FileUpload.fla` nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, scegliere *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/StageSize*.

Per informazioni su ciascun metodo, proprietà ed evento dell'API `FileReference`, consultare `FileReference` (`flash.net.FileReference`) e `FileReferenceList` (`flash.net.FileReferenceList`) nella *Guida di riferimento di ActionScript 2.0*.

## Informazioni sulla funzionalità e sicurezza dell'API FileReference

Flash Player e l'API FileReference (vedere “[Informazioni sul caricamento e scaricamento dei file](#)” a pagina 700) supportano il caricamento e scaricamento dei file fino a 100 MB. L'API FileReference *non* permette all'applicazione Flash, che inizializza il trasferimento dei file, di eseguire le seguenti operazioni:

- Accedere al file caricato o scaricato
- Accedere al percorso del file sul computer dell'utente

Se si utilizza un server che richiede l'autenticazione, l'unica operazione potenzialmente efficace consiste nell'effettuare lo scaricamento del file con il plug-in di Flash Player per il browser. Il caricamento su tutti i player di Flash o lo scaricamento mediante un Flash player autonomo o esterno dà esito negativo sui server che richiedono l'autenticazione. Utilizzare i listener di eventi FileReference per determinare se le operazioni vengono completate correttamente e come gestire gli eventuali errori.

Sia il caricamento che lo scaricamento sono limitati al dominio del file SWF, compresi gli eventuali domini specificati da un file di criteri dei domini. È necessario collocare un file di criteri sul server nel caso il file SWF che inizializza il caricamento o scaricamento non abbia lo stesso dominio del server. Per ulteriori informazioni sui file di criteri validi tra più domini, consultare “[Informazioni su domini, sicurezza tra domini e file SWF](#)” a pagina 753.

Mentre le chiamate a `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()` sono in esecuzione, la riproduzione del file SWF viene messa in pausa nelle piattaforme seguenti: plug-in per browser di Flash Player per Mac OS X, Flash Player esterno per Macintosh e player autonomo Macintosh su Mac OS X 10.1 e versioni precedenti. L'esecuzione del file SWF continua su tutti i player Windows e nel Flash Player autonomo-Macintosh su Mac su OS 10.2 e versioni successive.



Quando si permette agli utenti di caricare file su un server, è necessario sempre prestare attenzione a controllare il tipo di file prima di salvarlo sul disco rigido. Se, ad esempio, non si desidera permettere ad un utente di caricare uno script sul lato server che potrebbe essere utilizzato per eliminare cartelle o file sul server. Nel caso si desideri permettere solo agli utenti di caricare un file immagine, assicurarsi che lo script sul lato server, che carica i file, controlli che il file in corso di caricamento sia un'immagine valida.

Per informazioni sull'applicazione dell'API FileReference, consultare “[Aggiunta della funzionalità di caricamento file ad un'applicazione](#)” a pagina 703.

## Aggiunta della funzionalità di caricamento file ad un'applicazione

La procedura seguente mostra come costruire un'applicazione che consenta di caricare file immagine su un server. L'applicazione permette agli utenti di selezionare un'immagine da caricare sul proprio disco rigido e quindi inviarla al server. L'immagine caricata viene visualizzata quindi nel file SWF utilizzato per caricare l'immagine.

L'esempio che costruisce l'applicazione Flash è un esempio che descrive nel dettaglio il codice sul lato server. È necessario ricordare che i file immagine sono limitati per dimensioni: possibile caricare solo immagini con una dimensione non superiore a 200 Kb.

### Per costruire un'applicazione FLA con l'API FileReference:

1. Creare un nuovo documento Flash e salvarlo come **fileref.fla**.
2. Aprire il pannello Componenti, trascinare il componente ScrollPane sullo stage e assegnare ad esso un nome di istanza di **imagePane**. L'istanza ScrollPane viene dimensionata e riposizionata in un secondo momento mediante ActionScript.
3. Trascinare un componente Button nello stage e assegnare ad esso il nome di istanza **uploadBtn**.
4. Trascinare due componenti Label sullo stage ed assegnare loro i nomi di istanza **imageLbl** e **statusLbl**.
5. Trascinare un componente ComboBox nello stage e assegnare ad esso il nome di istanza **imagesCb**.
6. Trascinare un componente TextArea nello stage e assegnare ad esso il nome di istanza **statusArea**.
7. Creare un nuovo simbolo per clip filmato sullo Stage ed aprire il simbolo per modificarlo (fare doppio clic sull'istanza per aprirlo in modalità di modifica dei simboli).
8. Creare un nuovo campo testo statico all'interno del clip filmato, quindi aggiungere il testo seguente:

#### **Il file che si è tentato di scaricare non è nel server.**

Nell'applicazione finale, questa avvertenza potrebbe apparire per una delle ragioni seguenti:

- L'immagine è stata eliminata dalla coda sul server quando sono state caricate altre immagini.
- Il server non ha copiato l'immagine perché superava i 200 KB.

- Il tipo di file non era un file JPEG, GIF o PNG valido.

**NOTA**

La larghezza del campo testo deve essere inferiore a quella dell'istanza ScrollPane (400 pixel), altrimenti gli utenti dovranno scorrere il messaggio di errore in orizzontale per visualizzarlo.

9. In Libreria, fare clic con il pulsante destro del mouse sul simbolo, quindi selezionare Concatenamento dal menu di scelta rapida.
10. Selezionare le caselle di controllo Esporta per ActionScript ed Esporta nel primo fotogramma e digitare **Messaggio** nella casella di testo Identificatore. Fare clic su OK.
11. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale:

**NOTA**

I commenti del codice comprendono i dettagli relativi alla funzionalità. Una panoramica del codice segue l'esempio.

```
import flash.net.FileReference;

imagePane.setSize(400, 350);
imagePane.move(75, 25);
uploadBtn.move(75, 390);
uploadBtn.label = "Upload Image";
imageLbl.move(75, 430);
imageLbl.text = "Select Image";
statusLbl.move(210, 390);
statusLbl.text = "Status";
imagesCb.move(75, 450);
statusArea.setSize(250, 100);
statusArea.move(210, 410);

/* L'oggetto listener "ascolta" l'evento FileReference. */
var listener:Object = new Object();

/* Quando l'utente seleziona un file, viene richiamato il metodo
 onSelect() e passato un riferimento all'oggetto FileReference. */
listener.onSelect = function(selectedFile:FileReference):Void {
 /* Aggiornare la TextArea per notificare all'utente che Flash sta
 tentando di caricare l'immagine. */
 statusArea.text += "Attempting to upload " + selectedFile.name + "\n";
 /* Aggiornare il file allo script PHP sul server. */
 selectedFile.upload("http://www.helpexamples.com/flash/file_io/
 uploadFile.php");
};

/* Quando inizia il caricamento del file, viene richiamato il metodo
 onOpen(), in modo da notificare all'utente che il caricamento sta
 iniziando. */
listener.onOpen = function(selectedFile:FileReference):Void {
 statusArea.text += "Opening " + selectedFile.name + "\n";
```

```

};

/* Una volta caricato il file, viene richiamato il metodo onComplete().
 */
listener.onComplete = function(selectedFile:FileReference):Void {
 /* Notificare all'utente che Flash sta iniziando a scaricare
 l'immagine. */
 statusArea.text += "Downloading " + selectedFile.name + " to
 player\n";
 /* Aggiungere l'immagine al componente ComboBox. */
 imagesCb.addItem(selectedFile.name);
 /* Impostare l'indice selezionato del ComboBox su quello dell'ultima
 immagine aggiunta. */
 imagesCb.selectedIndex = imagesCb.length - 1;
 /* Chiamare la funzione personalizzata downloadImage(). */
 downloadImage();
};

var imageFile:FileReference = new FileReference();
imageFile.addListener(listener);

imagePane.addEventListener("complete", imageDownloaded);
imagesCb.addEventListener("change", downloadImage);
uploadBtn.addEventListener("click", uploadImage);

/* Se l'immagine non viene scaricata, la proprietà totale dell'oggetto
 evento sarà uguale a -1. In tal caso, visualizzare un messaggio per
 l'utente. */
function imageDownloaded(event:Object):Void {
 if (event.total == -1) {
 imagePane.contentPath = "Message";
 }
}

/* Quando l'utente seleziona un'immagine dal ComboBox o quando si chiama
 la funzione downloadImage() direttamente dal metodo
 listener.onComplete(), la funzione downloadImage() imposta il
 contentPath del ScrollPane per avviare lo scaricamento dell'immagine
 al player. */
function downloadImage(event:Object):Void {
 imagePane.contentPath = "http://www.helpexamples.com/flash/file_io/
 images/" + imagesCb.value;
}

/* Quando l'utente fa clic sul pulsante, Flash chiama la funzione
 uploadImage() e apre una finestra di dialogo per cercare i file. */
function uploadImage(event:Object):Void {
 imageFile.browse([{description: "Image Files", extension:
 "*.*.jpg;*.gif;*.png"}]);
}

```

Questo codice ActionScript importa prima la classe FileReference, quindi inizializza, posiziona e ridimensiona ciascun componente nello stage. Successivamente, viene definito un oggetto listener, quindi tre gestori di eventi: onSelect, onOpen e onComplete.

L'oggetto listener viene aggiunto al nuovo oggetto FileReference denominato imageFile. Quindi vengono aggiunti i listener di eventi all'istanza ScrollPane imagePane, all'istanza ComboBox imagesCb e all'istanza Button uploadBtn. Ciascuna funzione dei listener di eventi viene definita nel codice che segue questa sezione di codice.

La prima funzione, imageDownloaded(), verifica che la quantità dei byte totali delle immagini scaricate sia -1 e, in caso affermativo, imposta il contentPath per l'istanza ScrollPane al clip filmato con l'identificatore di concatenamento di Messaggio creato in una fase precedente. La seconda funzione, downloadImage(), cerca di scaricare l'ultima immagine scaricata nell'istanza ScrollPane. Una volta scaricata l'immagine, viene attivata la funzione imageDownloaded() definita in precedenza che verifica se lo scaricamento sia avvenuto correttamente. La funzione finale, uploadImage(), apre una finestra di dialogo per la ricerca dei file che filtra tutte le immagini in formato JPEG, GIF e PNG.

12. Salvare le modifiche apportate al documento.
  13. Selezionare le impostazioni File > Pubblica, quindi la scheda Formati ed assicurarsi che sia selezionato sia Flash sia HTML.
  14. (Opzionale) Nella finestra di dialogo Impostazioni pubblicazione, selezionare la scheda Flash, quindi l'opzione Accedi solo alla rete dal menu a comparsa Sicurezza riproduzione locale.
- Se si completa questa fase, non sarà possibile eseguire delle restrizioni di sicurezza qualora si provi il documento in un browser locale.
15. Nella finestra di dialogo Impostazioni pubblicazione, fare clic su Pubblica per creare file HTML e SWF.

Al termine, passare alla procedura successiva in cui creare il contenitore del file SWF.

È possibile trovare i file di origine di questo esempio, FileUpload.fla, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, scegliere *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/StageSize*.

La procedura seguente richiede che sia installato PHP sul server web e che si disponga delle autorizzazioni di scrittura alle sottocartelle denominate immagini e temporanee. È necessario innanzitutto completare la procedura precedente o utilizzare il file SWF terminato disponibile nelle cartelle precedentemente menzionate.

**Per creare uno script sul lato server per l'applicazione di caricamento dell'immagine:**

1. Creare un nuovo documento PHP mediante un editor di testo come Dreamweaver o Notepad.
2. Aggiungere il seguente codice di PHP al documento. Una panoramica del codice segue l'esempio.

```
<?php

$MAXIMUM_FILESIZE = 1024 * 200; // 200KB
$MAXIMUM_FILE_COUNT = 10; // mantenere massimo 10 file su server
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE) {
 move_uploaded_file($_FILES['Filedata']['tmp_name'], "./temporary/
 ".$_FILES['Filedata']['name']);
 $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
 if ($type == 1 || $type == 2 || $type == 3) {
 rename("./temporary/".$_FILES['Filedata']['name'], "./images/
 ".$_FILES['Filedata']['name']);
 } else {
 unlink("./temporary/".$_FILES['Filedata']['name']);
 }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory)) {
 array_push($files, array('./images/'.$file, filectime('./images/
 '.$file)));
}
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT) {
 $files_to_delete = array_splice($files, 0, count($files) -
 $MAXIMUM_FILE_COUNT);
 for ($i = 0; $i < count($files_to_delete); $i++) {
 unlink($files_to_delete[$i][0]);
 }
}
print_r($files);
closedir($directory);

function sorter($a, $b) {
 if ($a[1] == $b[1]) {
 return 0;
 } else {
 return ($a[1] < $b[1]) ? -1 : 1;
 }
}
?>
```

Questo codice PHP definisce prima due variabili costanti: `$MAXIMUM_FILESIZE` e `$MAXIMUM_FILE_COUNT`. Tali variabili impongono la dimensione massima, in kilobyte, di un'immagine caricata sul server (200 KB), nonché come poter mantenere gli ultimi file caricati nella cartella delle immagini (10). Se la dimensione del file dell'immagine in corso di caricamento è inferiore o uguale al valore di `$MAXIMUM_FILESIZE`, l'immagine verrà spostata nella cartella temporanea.

Scessivamente, viene controllato il tipo di file caricato per assicurarsi che l'immagine sia in formato JPEG, GIF o PNG. Se l'immagine è un tipo compatibile, viene copiata dalla cartella temporanea alla cartella delle immagini. Qualora il file caricato non fosse uno dei tipi di immagine consentiti, viene eliminato dal file system.

Quindi viene creato un elenco di directory della cartella delle immagini su cui si eseguono elaborazioni cicliche mediante un ciclo while. Tutti i file nella cartella delle immagini vengono aggiunti ad un array e quindi ordinati. Se il numero corrente di file nella cartella delle immagini è superiore al valore di `$MAXIMUM_FILE_COUNT`, i file vengono eliminati finché non restano solo immagini `$MAXIMUM_FILE_COUNT`. In questo modo si evita che la cartella delle immagini raggiunga una dimensione ingestibile, poiché la cartella può contenere solo 10 immagini per volta e ciascuna immagine non deve superare i 200 KB (circa 2 MB di immagini per volta).

3. Salvare le modifiche al documento PHP.
4. Caricare i file SWF, HTML e PHP sul server Web.
5. Visualizzare il documento HTML remoto in un browser Web, quindi fare clic sul pulsante Carica immagine.
6. Individuare un file immagine sul disco rigido, quindi selezionare Apri dalla finestra di dialogo.

Il file SWF carica il file immagine nel documento PHP remoto e lo visualizza nello ScrollPane, che aggiunge barre di scorrimento, se necessario. Per visualizzare un'immagine scaricata in precedenza, selezionare il nome del file dall'istanza ComboBox nello stage. Se l'utente cerca di caricare un'immagine il cui tipo non è consentito (solo immagini in formato JPEG, GIF o PNG) o la dimensione del file è troppo grande (oltre i 200 KB), Flash visualizza il messaggio di errore nel clip filmato Messaggio della Libreria.

È possibile trovare i file di origine di questo esempio, FileUpload.fla, nella cartella Samples sul disco rigido.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters*.
- In Macintosh, scegliere *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/StageSize*.

Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

Per ulteriori informazioni sulla scrittura PHP, consultare il sito web [www.php.net/](http://www.php.net/).

## Informazioni su XML

Il linguaggio *XML* (Extensible Markup Language) sta diventando lo standard per lo scambio di dati strutturati nelle applicazioni Internet. I dati di Flash possono essere integrati con server che si avvalgono della tecnologia XML per creare applicazioni sofisticate, ad esempio programmi chat o sistemi di quotazioni azionarie.

XML, come HTML, utilizza tag per specificare o *contrassegnare* il corpo di un testo. In HTML si utilizzano tag predefiniti per indicare l'aspetto del testo in un browser, ad esempio il tag **<b>** per indicare il grassetto. In XML, invece, si definiscono tag che identificano il tipo di dati, ad esempio **<password>VerySecret</password>**. Il linguaggio XML separa la struttura delle informazioni dal loro aspetto, pertanto lo stesso documento XML può essere riutilizzato in ambienti diversi.

Ogni tag XML è detto *nodo* o elemento. Ogni nodo, a sua volta, dispone di un tipo (1 che indica un elemento XML o 3 che indica un nodo di testo); gli elementi possono anche disporre di attributi. Un nodo nidificato in un altro nodo è detto *nodo secondario*. La struttura gerarchica dei nodi è detta DOM XML ed è analoga alla struttura DOM JavaScript, ovvero alla struttura degli elementi in un browser.

Nell'esempio seguente, **<portfolio>** è il nodo principale. Non dispone di attributi e contiene il nodo secondario **<holding>** che invece ha gli attributi *symbol*, *qty*, *price* e *value*:

```
<portfolio>
 <holding symbol="rich"
 qty="75"
 price="245.50"
 value="18412.50" />
</portfolio>
```

Per ulteriori informazioni, consultare i seguenti argomenti:

- “[Uso della classe XML](#)” a pagina 711
- “[Uso della classe XMLSocket](#)” a pagina 717

Per ulteriori informazioni su XML, consultare il sito Web [www.w3.org/XML](http://www.w3.org/XML).

Sul disco rigido sono presenti molti file di esempio che caricano XML in un file SWF in fase di runtime. Un esempio dimostra come creare un Web log tracker caricando, analizzando e manipolando i dati XML. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `xml_blogTracker.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_BlogTracker*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/XML\_BlogTracker*.

Un secondo esempio dimostra come utilizzare XML e array nidificati per la selezione di stringhe in diverse lingue che compilino campi di testo. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `xml_languagePicker.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_LanguagePicker*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/XML\_LanguagePicker*.

Un terzo esempio dimostra come creare un menu dinamico con dati XML. Il campione chiama la funzione di costruzione `XmlMenu()` di ActionScript e passa ad essa due parametri: il percorso del file di menu XML e un riferimento alla linea temporale corrente. Il resto della funzionalità risiede nel file di classe personalizzato `XmlMenu.as`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `xmlmenu.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_Menu*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/XML\_Menu*.

## Uso della classe XML

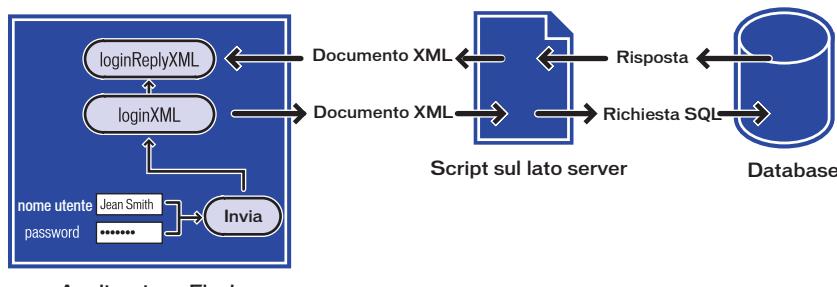
I metodi della classe XML ActionScript, ad esempio `appendChild()`, `removeNode()` e `insertBefore()`, consentono di strutturare i dati XML in Flash per inviarli a un server e gestire e interpretare i dati XML scaricati.

I seguenti metodi della classe XML inviano e caricano dati XML su un server tramite il metodo POST HTTP:

- Il metodo `load()` scarica i dati XML da un URL e li inserisce in un oggetto XML ActionScript.
- Il metodo `send()` codifica l'oggetto XML in un documento XML e lo invia a un URL specificato tramite il metodo POST. Se specificato, i dati restituiti vengono visualizzati in una finestra del browser.
- Il metodo `sendAndLoad()` invia un oggetto XML a un URL. Le informazioni restituite vengono inserite in un oggetto XML ActionScript.

È possibile, ad esempio, creare un sistema di quotazioni azionarie che memorizzi tutte le informazioni (nomi utente, password, ID di sessione, investimenti e informazioni sulle transazioni) in un database.

Lo script sul lato server che passa le informazioni tra Flash e il database legge e scrive i dati in formato XML. È possibile utilizzare ActionScript per convertire le informazioni raccolte nel file SWF (ad esempio un nome utente e una password) in un oggetto XML e quindi inviare i dati allo script sul lato server come documento XML, oppure per caricare il documento XML che il server restituisce in un oggetto XML da utilizzare nel file SWF.



*Flusso e conversione dei dati tra un file SWF, uno script sul lato server e un database*

La convalida della password per il sistema di quotazioni azionarie richiede due script: una funzione definita sul fotogramma 1 e uno script che crea e invia gli oggetti XML creati nel documento.

Quando un utente immette informazioni nei campi di testo del file SWF con le variabili `username` e `password`, le variabili devono essere convertite in formato XML prima di essere passate al server. La prima sezione dello script carica le variabili in un nuovo oggetto XML denominato `loginXML`. Quando un utente fa clic su un pulsante per eseguire l'accesso, l'oggetto `loginXML` viene convertito in una stringa XML e inviato al server.

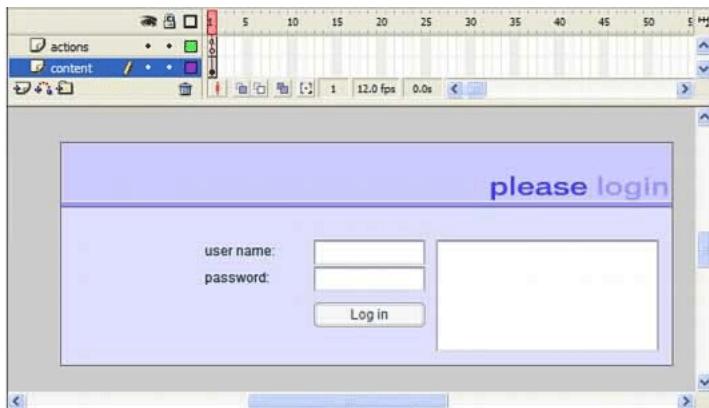
Il codice ActionScript seguente viene inserito nella linea temporale ed utilizzato per inviare dati in formato XML al server. Per comprendere lo script, leggere le righe di commento che iniziano con //:

```
// Ignora gli spazi vuoti nel codice XML
XML.prototype.ignoreWhite = true;
// Crea un oggetto XML in cui inserire la risposta del server
var loginReplyXML:XML = new XML();
// Questa funzione viene attivata alla ricezione di un pacchetto XML dal
// server.
loginReplyXML.onLoad = function(success:Boolean) {
 if (success) {
 // (Opzionale) Crea due campi di testo per lo stato e il debug
 // status_txt.text = this.firstChild.attributes.status;
 // debug_txt.text = this.firstChild;
 switch (this.firstChild.attributes.STATUS) {
 case 'OK' :
 _global.session = this.firstChild.attributes.SESSION;
 trace(_global.session);
 gotoAndStop("welcome");
 break;
 case 'FAILURE' :
 gotoAndStop("loginfailure");
 break;
 default :
 // Questo non deve mai accadere
 trace("Ricevuto valore imprevisto per STATUS.");
 }
 } else {
 trace("Si è verificato un errore.");
 }
};
// Questa funzione viene attivata quando si fa clic su login_btn
login_btn.onRelease = function() {
 var loginXML:XML = new XML();
 // Crea dati in formato XML da inviare al server
 var loginElement:XMLNode = loginXML.createElement("login");
 loginElement.attributes.username = username_txt.text;
 loginElement.attributes.password = password_txt.text;
 loginXML.appendChild(loginElement);
 // Invia i dati in formato XML al server
```

```

 loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm",
 loginReplyXML);
};


```



Il test del codice può essere eseguito utilizzando il nome utente JeanSmith e la password VerySecret. La prima sezione dello script genera il seguente codice XML quando l'utente fa clic sul pulsante per eseguire l'accesso:

```
<login username="JeanSmith" password="VerySecret" />
```

Il server riceve il codice XML, genera una risposta XML e la invia nuovamente al file SWF. Se la password viene accettata, il server invia la seguente risposta:

```
<LOGINREPLY STATUS="OK" SESSION="4D968511" />
```

Questo codice XML include un attributo session contenente un ID di sessione univoco che viene utilizzato in tutte le comunicazioni tra il client e il server per tutta la durata della sessione. Se la password viene rifiutata, il server invia il seguente messaggio:

```
<LOGINREPLY STATUS="FAILURE" />
```

Il nodo XML loginreply deve essere caricato in un oggetto XML vuoto nel file SWF. L'istruzione seguente crea l'oggetto XML loginreplyXML per ricevere il nodo XML:

```
// Crea un oggetto XML in cui inserire la risposta del server
var loginReplyXML:XML = new XML();
loginReplyXML.onLoad = function(success:Boolean) {
```

La seconda istruzione definisce una funzione in linea anonima che viene chiamata all'attivazione dell'evento onLoad.

Il pulsante per l'accesso (istanza `login_btn`) consente di inviare il nome utente e la password in formato XML al server e di caricare una risposta XML nel file SWF. A questo scopo è possibile fare ricorso al metodo `sendAndLoad()`, come nell'esempio seguente:

```
loginXML.sendAndLoad("http://www.flash-mx.com.com/mm/main.cfm",
 loginReplyXML);
```

In primo luogo vengono creati i dati in formato XML utilizzando i valori che l'utente immette nel file SWF, quindi l'oggetto XML viene inviato con l'ausilio del metodo `sendAndLoad`. Analogamente a quanto avviene con i dati di una funzione `loadVariables()`, l'elemento XML `loginreply` funziona in modo asincrono, vale a dire non attende i risultati prima di essere restituito, e viene caricato nell'oggetto `loginReplyXML`. Alla ricezione dei dati, viene chiamato il gestore `onLoad` dell'oggetto `loginReplyXML`. È necessario definire la funzione `loginReplyXML` che viene chiamata all'attivazione del gestore `onLoad`, affinché possa elaborare l'elemento `loginreply`.

**NOTA**

Questa funzione deve essere contenuta sempre nel fotogramma che contiene il codice ActionScript relativo al pulsante di accesso.

Se l'accesso viene eseguito correttamente, il file SWF avanza all'etichetta del fotogramma `welcome`. In caso contrario, l'indicatore di riproduzione si sposta all'etichetta del fotogramma `loginfailure`. L'elaborazione avviene tramite una condizione e un'istruzione `case`. Per ulteriori informazioni sulle istruzioni `case` e `break`, vedere `case statement` e `break statement` e nella *Guida di riferimento di ActionScript 2.0*. Per ulteriori informazioni sulle condizioni, vedere `if statement` e `else statement` nella *Guida di riferimento di ActionScript 2.0*.

**NOTA**

Il codice riportato in precedenza viene fornito solo a scopo di esempio e Macromedia non ne garantisce il livello di sicurezza. Per implementare un sistema protetto da password è necessario avere familiarità con le tecniche atte a garantire la sicurezza di rete.

Per ulteriori informazioni, vedere Integrazione tra XML e Flash in un'applicazione Web all'indirizzo [www.macromedia.com/support/flash/interactivity/xml/](http://www.macromedia.com/support/flash/interactivity/xml/) e la voce XML nella *Guida di riferimento di ActionScript 2.0*. Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

Nella cartella Samples presente sul disco rigido è possibile trovare un file sorgente di esempio, `login.fla`. Questo campione mostra come aggiungere una semplice funzionalità di login ai propri siti Web mediante ActionScript 2.0; utilizza ActionScript e componenti per creare un piccolo modulo in cui inserire il nome utente e la password e fare clic su un pulsante per inserire il sito.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript>Login*.

- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Login*.

Flash Player 8 ha introdotto il gestore di eventi `onHTTPStatus` per la classe XML, la classe LoadVars e la classe MovieClipLoader per permettere agli utenti di accedere al codice di stato da una richiesta HTTP. Questa funzionalità consente agli sviluppatori di determinare il *perché* una particolare operazione di caricamento potrebbe non aver dato esito positivo piuttosto che determinare solo che tale operazione ha già dato esito negativo.

L'esempio seguente mostra come poter utilizzare il gestore eventi `onHTTPStatus` della classe XML per controllare se un file di testo è stato scaricato dal server e quale codice di stato è stato restituito dalla richiesta HTTP.

#### **Verifica dei codici di stato HTTP mediante la classe XML:**

1. Creare un nuovo documento Flash e salvarlo come `xmlhttp.fla`.
2. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onHTTPStatus = function(httpStatus:Number) {
 trace("HTTP status is: " + httpStatus);
};
my_xml.onLoad = function(success:Boolean) {
 if (success) {
 trace("Caricamento XML completato");
 // 0 (No error; parse was completed successfully.)
 trace("Stato XML: " + my_xml.status);
 } else {
 trace("Impossibile caricare XML");
 }
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

Il codice precedente definisce un nuovo oggetto XML con il nome della variabile `my_xml`, definisce due gestori di eventi (`onHTTPStatus` e `onLoad`) e carica un file XML esterno. Il gestore di eventi `onLoad` verifica se il file XML è stato correttamente caricato e, in caso affermativo, invia un messaggio al pannello Output, nonché traccia la proprietà di stato dell'oggetto XML. È importante ricordare che il listener di eventi `onHTTPStatus` restituisce il codice di stato restituito dal server Web, mentre la proprietà `XML.status` contiene un valore numerico che indica se è stato possibile riuscire ad analizzare l'oggetto XML.

**3.** Selezionare Controllo > Prova filmato per provare il documento Flash.

**SUGGERIMENTO**

Il gestore di eventi XML.onHTTPStatus è una nuova funzionalità di Flash Player 8.

**AVVISO**

Non confondere i codici httpStatus di HTPPP con la proprietà status della classe XML. Il gestore di eventi onHTTPStatus restituisce il codice di stato del server derivato da una richiesta HTPP e la proprietà status impostati automaticamente, nonché restituisce un valore numerico che indica se un documento XML è stato analizzato in un oggetto XML.

**ATTENZIONE**

Se un server Web non restituisce un codice status a Flash Player, al gestore di eventi onHTTPStatus viene restituito il numero 0.

Sul disco rigido sono presenti molti file di esempio che caricano XML in un file SWF in fase di runtime. Un esempio dimostra come creare un Web log tracker caricando, analizzando e manipolando i dati XML. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, *xml\_blogTracker.fla*.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_BlogTracker*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML\_BlogTracker*.

Un secondo esempio dimostra come utilizzare XML e array nidificati per la selezione di stringhe in diverse lingue che compilino campi di testo. Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, *xml\_languagePicker.fla*.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_LanguagePicker*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML\_LanguagePicker*.

Un terzo esempio dimostra come creare un menu dinamico con dati XML. Il campione chiama la funzione di costruzione `XmlMenu()` di ActionScript e passa ad essa due parametri: il percorso del file di menu XML e un riferimento alla linea temporale corrente. Il resto della funzionalità risiede nel file di classe personalizzato `XmlMenu.as`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `xmlmenu.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML\_Menu*.
- Su Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/XML\_Menu*.

## Uso della classe XMLSocket

ActionScript fornisce una classe `XMLSocket` incorporata che consente di stabilire una connessione continua con un server. Con una connessione socket il server può pubblicare o eseguire il *push* delle informazioni al client nel momento in cui queste sono disponibili. Se la connessione non è continua, il server deve attendere una richiesta HTTP. La connessione aperta evita problemi di latenza e viene in genere utilizzata per applicazioni in tempo reale, ad esempio per la conversazione in linea. I dati inviati sulla connessione socket sono formattati come stringa XML singola. Per strutturarli è possibile utilizzare la classe XML.

Per creare una connessione socket, è necessario creare un'applicazione sul lato server che attenda la richiesta della connessione socket e invii la risposta al file SWF. Questo tipo di applicazione sul lato server può essere scritta in un linguaggio di programmazione come Java.

**NOTA**

La classe `XMLSocket` non può eseguire il tunneling automatico attraverso i firewall perché, a differenza del protocollo RTMP (Real-Time Messaging Protocol), `XMLSocket` non dispone di capacità di tunneling HTTP. Se è necessario il tunneling HTTP, valutare la possibilità di utilizzare Flash Remoting o Flash Communication Server (che supporta RTMP).

Per trasferire il codice XML verso e dal server su una connessione socket, è possibile utilizzare i metodi `connect()` e `send()` della classe `XMLSocket`. Il metodo `connect()` stabilisce una connessione socket con una porta di un server Web. Il metodo `send()` passa un oggetto XML al server specificato nella connessione socket.

Quando si richiama il metodo `connect()`, Flash Player apre una connessione TCP/IP al server e la mantiene aperta fino al verificarsi di uno degli eventi seguenti:

- Viene chiamato il metodo `close()` della classe `XMLSocket`.
- Non sono più presenti riferimenti all'oggetto `XMLSocket`.

- Flash Player viene chiuso.
- La connessione viene interrotta, ad esempio il modem si disconnette.

Nell'esempio seguente viene creata una connessione socket XML e vengono inviati dati dall'oggetto XML myXML. Per comprendere lo script, leggere le righe di commento che iniziano con //:

```
// Creare l'oggetto XMLSocket
var theSocket:XMLSocket = new XMLSocket();
// Eseguire la connessione a un sito su una porta non utilizzata superiore a
// 1024 con l'ausilio del metodo connect()
// Immettere localhost o 127.0.0.1 per eseguire una prova locale
// Per un server esistente, immettere il proprio dominio nel formato
// www.dominio.com
theSocket.connect("localhost", 12345);
// Visualizza testo relativo alla connessione
theSocket.onConnect = function(myStatus) {
 if (myStatus) {
 conn_txt.text = "connection successful";
 } else {
 conn_txt.text = "no connection made";
 }
};
// Dati da inviare
function sendData() {
 var myXML:XML = new XML();
 var mySend = myXML.createElement("thenode");
 mySend.attributes.myData = "someData";
 myXML.appendChild(mySend);
 theSocket.send(myXML);
}
// Facendo clic sul pulsante i dati vengono inviati
sendButton.onRelease = function() {
 sendData();
};
// Analizza i dati restituiti dalla connessione socket
theSocket.onData = function(msg:String):Void {
 trace(msg);
};
```

Per ulteriori informazioni, vedere la voce XMLSocket nella *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

# Invio di messaggi verso e da Flash Player

Per inviare messaggi da un file SWF al relativo ambiente host, ad esempio un browser Web, un filmato di Macromedia Director o la versione autonoma di Flash Player, si può utilizzare la funzione `fscommand()` che permette di estendere il file SWF utilizzando le funzionalità dell'host. È possibile, ad esempio, passare una funzione `fscommand()` a una funzione JavaScript in una pagina HTML per aprire una nuova finestra del browser con determinate proprietà.

Per controllare un file SWF in Flash Player da linguaggi di script del browser come JavaScript, VBScript e Microsoft JScript, è possibile avvalersi di metodi di Flash Player, ovvero funzioni che inviano messaggi da un ambiente host al file SWF. In una pagina HTML, ad esempio, è possibile inserire un collegamento che invii il file SWF a un fotogramma specifico.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “Uso della funzione `fscommand()`” a pagina 719
- “Informazioni sull'uso di JavaScript per controllare le applicazioni Flash” a pagina 722
- “Informazioni sui metodi di Flash Player” a pagina 722

## Uso della funzione `fscommand()`

### NOTA

L'API External sostituisce `fscommand()` in Flash 8 per interoperare con una pagina HTML o un'applicazione contenitore. In questa situazione l'API External offre una funzionalità più efficace rispetto a `fscommand()`. Per ulteriori informazioni, vedere “[Informazioni sull'API External](#)” a pagina 723.

La funzione `fscommand()` consente di inviare un messaggio al programma che ospita Flash Player, ad esempio un browser Web.

### NOTA

`fscommand()` non può essere utilizzato per chiamare JavaScript in Safari o Internet Explorer per Macintosh.

La funzione dispone di due parametri: *comando* e *argomenti*. Per inviare un messaggio alla versione autonoma di Flash Player, è necessario utilizzare comandi e argomenti predefiniti. Il gestore di eventi seguente, ad esempio, imposta il lettore autonomo affinché ridimensioni il file SWF a schermo intero quando viene rilasciato il pulsante:

```
my_btn.onRelease = function() {
 fscommand("fullscreen", true);
};
```

Nella tabella seguente sono riportati i valori che possono essere specificati per i parametri *comando* e *argomenti* di `fscommand()` per controllare la riproduzione e l'aspetto di un file SWF all'interno del lettore autonomo, compresi i proiettori.

**NOTA**

Un proiettore è un file SWF salvato in un formato che può essere eseguito come applicazione autonoma, cioè incorporando Flash Player con il contenuto di un file eseguibile.

Comando	Argomenti	Scopo
quit	Nessuno	Chiude il proiettore.
fullscreen	true o false	Se si specifica true, Flash Player viene impostato in modalità a schermo intero. Se si specifica false il lettore viene riportato alla normale visualizzazione con i menu.
allowscale	true o false	Se si specifica false, il lettore viene impostato affinché il file SWF sia disegnato sempre alle dimensioni originali e non venga mai ridimensionato. Se si specifica true, il file SWF viene ridimensionato per essere adattato alle dimensioni del lettore.
showmenu	true o false	Se si specifica true, vengono attivate tutte le voci del menu di scelta rapida. Se si specifica false, vengono disattivate tutte le voci del menu di scelta rapida ad eccezione di Impostazioni e Informazioni su Flash.
exec	Percorso dell'applicazione	Esegue un'applicazione dall'interno del proiettore.

Per utilizzare `fscommand()` per inviare un messaggio a un linguaggio di script come JavaScript in un browser, è possibile passare qualunque coppia di parametri *comando* e *argomenti*, che possono essere stringhe o espressioni e vengono utilizzati in una funzione JavaScript che provvederà a "catturare", ovvero a gestire, la funzione `fscommand()`.

Una funzione `fscommand()` richiama la funzione JavaScript `nameofilmato_DoFSCommand` nella pagina HTML in cui è presente il file SWF. `nameofilmato` è il nome di Flash Player assegnato dall'attributo `name` del tag `embed` o dall'attributo `id` del tag `object`. Se al file SWF viene assegnato il nome `myMovie`, la funzione JavaScript richiamata è `myMovie_DoFSCommand`.

### Per utilizzare `fscommand()` per aprire una finestra di messaggio da un file SWF nella pagina HTML tramite JavaScript:

1. Creare un nuovo file FLA e salvarlo come `myMovie.fla`.
2. Trascinare due istanze del componente Button sullo stage e assegnare i nomi di istanza `window_btn` e `alert_btn` e, rispettivamente, le etichette `Apri finestra` e `Avviso`.
3. Inserire un nuovo livello sulla linea temporale e rinominarlo **Azioni**.

- 4.** Selezionare il fotogramma 1 del livello Azioni e aggiungere il seguente codice ActionScript nel pannello Azioni:

```
window_btn.onRelease = function() {
 fscommand("popup", "http://www.macromedia.com/");
};

alert_btn.onRelease = function() {
 fscommand("alert", "You clicked the button.");
};
```

- 5.** Selezionare File > Impostazioni pubblicazione e verificare che nella casella di riepilogo Modello della scheda HTML sia selezionato Flash con FSCommand.
- 6.** Selezionare File > Pubblica per generare i file SWF e HTML.
- 7.** In un editor di testo o HTML, aprire il file HTML generato al punto 6 ed esaminare il codice. Se il file SWF viene pubblicato selezionando il modello Flash con FSCommand nella scheda HTML della finestra di dialogo Impostazioni pubblicazione, nel file HTML viene aggiunto ulteriore codice. Gli attributi NAME e ID del file SWF corrispondono al nome del file, ad esempio, per il file myMovie.fla, gli attributi verrebbero impostati su myMovie.
- 8.** Nel file HTML, aggiungere il seguente codice JavaScript nel punto in cui è presente il commento // Place your code here.:

```
if (command == "alert") {
 alert(args);
} else if (command == "popup") {
 window.open(args, "mmwin", "width=500,height=300");
}
```

Per ulteriori informazioni sulla pubblicazione, consultare [Capitolo 17, “Pubblicazione”](#) in *Uso di Flash*.

In alternativa, nelle applicazioni per Microsoft Internet Explorer, è possibile associare un gestore di eventi direttamente nel tag <SCRIPT>, come nell'esempio seguente:

```
<script Language="JavaScript" event="FSCommand (command, args)"
for="theMovie">
...
</script>
```

- 9.** Salvare e chiudere il file HTML.

Quando si modificano file HTML all'esterno di Flash, ricordarsi di deselezionare la casella di controllo HTML in File > Impostazioni pubblicazione. In caso contrario, il codice HTML viene sovrascritto da Flash durante la pubblicazione successiva.

- 10.** Aprire il file HTML in un browser per visualizzarlo. Fare clic sul pulsante Apri finestra. Viene aperta una finestra del browser con il sito Web Macromedia. Fare clic sul pulsante Avviso. Viene aperta una finestra di avviso.

Con la funzione `fscommand()` è possibile inviare messaggi a Macromedia Director che vengono interpretati da Lingo come stringhe, eventi o codice Lingo eseguibile. Se il messaggio è una stringa o un evento, è necessario scrivere codice Lingo per riceverlo dalla funzione `fscommand()` ed eseguire un'azione in Director. Per ulteriori informazioni, visitare il Centro di assistenza di Director all'indirizzo [www.macromedia.com/support/director](http://www.macromedia.com/support/director).

In Visual Basic, Visual C++ e altri programmi che possono ospitare controlli ActiveX, `fscommand()` invia un evento VB con due stringhe che possono essere gestite dal linguaggio di programmazione dell'ambiente. Per ulteriori informazioni, cercare le parole chiave *Flash method* nel Centro di supporto di Flash all'indirizzo [www.macromedia.com/support/flash](http://www.macromedia.com/support/flash/).

## Informazioni sull'uso di JavaScript per controllare le applicazioni Flash

Flash Player 6 (6.0.40.0) e le versioni successive supportano alcuni metodi JavaScript specifici per le applicazioni Flash e `FSCommand` in Netscape 6.2 e versioni successive. Le versioni precedenti non supportano questi metodi JavaScript né `FSCommand` in Netscape 6.2 o versioni successive. Per ulteriori informazioni, vedere l'articolo relativo alla creazione di script con Flash nel Centro di supporto Macromedia all'indirizzo [www.macromedia.com/support/flash/publishexport/scriptingwithflash/](http://www.macromedia.com/support/flash/publishexport/scriptingwithflash/).

Per Netscape 6.2 e versioni successive non è necessario impostare l'attributo `swliveconnect` su `true`, anche se questa impostazione non produce risultati negativi sul file SWF. Per ulteriori informazioni, vedere l'attributo `swLiveConnect` in “[Parametri e attributi](#)” a pagina 556 in *Uso di Flash*.

## Informazioni sui metodi di Flash Player

I metodi di Flash Player possono essere utilizzati per controllare un file SWF in Flash Player con linguaggi di script del browser come JavaScript e VBScript. Analogamente a quanto avviene con altri metodi, è possibile infatti utilizzare i metodi di Flash Player per effettuare chiamate a file SWF da un ambiente per la creazione di script diverso da ActionScript. Ogni metodo ha un nome e la maggior parte di essi accetta parametri. Un parametro specifica un valore in base al quale il metodo agisce. Il calcolo eseguito da alcuni metodi restituisce un valore che può essere utilizzato nell'ambiente di creazione degli script.

Due tecnologie attivano la comunicazione tra il browser e Flash Player: LiveConnect (Netscape Navigator 3.0 o versione successiva per Windows 95/98/2000/NT/XP o Power Macintosh) e ActiveX (Internet Explorer 3.0 e versione successiva per Windows 95/98/2000/NT/XP). Sebbene le tecniche per la creazione di script siano simili per tutti i browser e i linguaggi, per i controlli ActiveX sono disponibili ulteriori proprietà ed eventi.

Per ulteriori informazioni e per un elenco completo dei metodi per la creazione di script di Flash Player, cercare le parole chiave *Flash method* nel Centro di supporto di Flash all'indirizzo [www.macromedia.com/support/flash](http://www.macromedia.com/support/flash).

## Informazioni sull'API External

La classe ExternalInterface viene anche denominata l'*API External*, un nuovo sottosistema che consente facili comunicazioni tra ActionScript e il contenitore Flash Player e una pagina HTML contenente JavaScript o un'applicazione desktop che incorpora Flash Player.

**NOTA**

Questa funzionalità sostituisce la precedente funzione `fscommand()` per l'interoperatività con una pagina HTML o un'applicazione contenitore. In questa situazione l'API External offre una funzionalità più efficace rispetto a `fscommand()`. Per ulteriori informazioni, vedere ["Informazioni sull'API External" a pagina 723](#).

La classe ExternalInterface è disponibile solo nelle seguenti circostanze:

- In tutte le versioni supportate di Internet Explorer per Windows (5.0 e versioni successive)
- In un contenitore ActiveX personalizzato ed incorporato, come un'applicazione desktop che incorpora il controllo ActiveX di Flash Player.
- In tutti i browser che supportano l'interfaccia NPRuntime che comprende attualmente i seguenti browser:
  - Firefox 1.0 e versioni successive
  - Mozilla 1.7.5 e versioni successive
  - Netscape 8.0 e versioni successive
  - Safari 1.3 e versioni successive.

In tutte le altre situazioni in cui la proprietà `ExternalInterface.available` restituisce false.

Da ActionScript, è possibile chiamare una funzione JavaScript sulla pagina HTML. L'API External offre la seguente funzionalità perfezionata rispetto a `fscommand()`:

- È possibile utilizzare qualsiasi funzione JavaScript, non solo le funzioni utilizzabili con `fscommand function`.
- È possibile passare un numero qualsiasi di argomenti, con tutti i nomi; non ci sono limitazioni al passaggio di un comando e argomenti.

- È possibile passare diversi tipi di dati (come Booleano, Numero e Stringa); non ci sono più limitazioni per i parametri String.
- È possibile ora ricevere il valore di una chiamata e la restituzione immediata di tale valore ad ActionScript come valore restituito per la chiamata eseguita.

È possibile chiamare una funzione di ActionScript da JavaScript su una pagina HTML.

Per ulteriori informazioni, vedere `ExternalInterface`  
`(flash.external.ExternalInterface)`. Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina [735](#).

Le sezioni seguenti contengono esempi che utilizzano l'API External:

- “[Creazione di interazione con l'API External](#)” a pagina [724](#)
- “[Controllo di Flash Video con l'API External](#)” a pagina [727](#)

## Creazione di interazione con l'API External

È possibile creare interazione tra il browser e un file SWF incorporato in una pagina Web. La procedura seguente invia testo alla pagina HTML contenente il file SWF e HTML rinvia un messaggio al file SWF in fase di runtime.

### Per creare l'applicazione Flash:

1. Creare un nuovo documento Flash e salvarlo come `extint.fla`.
2. Trascinare due componenti `TextInput` sullo stage ed assegnare loro i nomi di istanza `in_ti` e `out_ti`.
3. Trascinare un componente `Label` sullo stage, assegnergli il nome dell'istanza di `out_lbl`, posizionarlo sopra l'istanza `TextInput` `out_ti` e impostare **Invio a JS** come proprietà di testo nella scheda Parametri della finestra di ispezione Proprietà.
4. Trascinare un componente `Button` nello stage, posizionarlo accanto all'etichetta `out_lbl` e assegnergli il nome dell'istanza `send_button`.
5. Trascinare un componente `Label` sullo stage, assegnergli il nome dell'istanza di `in_lbl`, posizionarlo sopra l'istanza `TextInput` `in_ti` e impostare **Ricezione da JS** come proprietà di testo nella scheda Parametri della finestra di ispezione Proprietà.
6. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
import flash.external.ExternalInterface;

ExternalInterface.addCallback("asFunc", this, asFunc);
function asFunc(str:String):Void {
 in_ti.text = "JS > Hello " + str;
}
```

```

send_button.addEventListener("click", clickListener);
function clickListener(eventObj:Object):Void {
 trace("click > " + out_ti.text);
 ExternalInterface.call("jsFunc", out_ti.text);
}

```

Il codice precedente viene suddiviso in tre sezioni. La prima sezione importa la classe ExternalInterface in modo da non dover utilizzare il nome completo di classe. La seconda sezione del codice definisce una funzione di callback, asFunc(), che viene chiamata da JavaScript in un documento HTML creato in un esempio in programma. Questa funzione imposta il testo all'interno del componente TextInput nello stage. La terza sezione di codice definisce una funzione e lo assegna come un listener di eventi per quando l'utente fa clic sull'istanza di componente Button nello stage. In qualsiasi momento si fa clic sul pulsante, il file SWF chiama la funzione JavaScript jsFunc() nella pagina HTML e passa la proprietà testo dell'istanza di input testo out\_ti.

7. Selezionare File > Impostazioni pubblicazione, quindi selezionare la scheda Formati ed assicurarsi che sia selezionato sia Flash sia HTML.
8. Fare clic su Pubblica per creare file HTML e SWF.

Al termine, passare alla procedura successiva per creare il contenitore del file SWF.

Prima di poter provare il documento Flash precedente, è necessario modificare il codice HTML generato ed aggiungere altro codice HTML e JavaScript. La procedura seguente modifica il contenitore HTML per il file SWF in modo che i due file possano interagire quando vengono eseguiti in un browser.

#### **Per creare il contenitore HTML per il file SWF:**

1. Completare la procedura precedente.
2. Aprire il file extint.html creato da Flsh quando si pubblica l'applicazione.

Si tratta della stessa cartella del documento Flash.

3. Aggiungere il codice JavaScript seguente tra i tag di apertura e chiusura head:

```

<script language="JavaScript">
<!--
function thisMovie(movieName) {
 var isIE = navigator.appName.indexOf("Microsoft") != -1;
 return (isIE) ? window[movieName] : document[movieName];
}

function makeCall(str) {
 thisMovie("extint").asFunc(str);
}

function jsFunc(str) {

```

```

 document.inForm.inField.value = "AS > Hello " + str;
 }
// -->
</script>
```

Questo codice JavaScript definisce tre metodi. Il primo restituisce un riferimento al file SWF incorporato a seconda che il browser dell'utente sia Microsoft Internet Explorer (IE) o Mozilla. La seconda funzione, `makeCall()`, chiama il metodo `asFunc()` definito nel documento Flash nell'esempio precedente. Il parametro "extint" nella chiamata di funzione `thisMovie()` fa riferimento all'ID dell'oggetto e incorpora il nome del file SWF incorporato. Se il documento Flash è stato salvato con un nome diverso, è necessario modificare questa stringa in modo che corrisponda ai valori nei tag `object` e `tag`. La terza funzione, `jsFunc()`, imposta il valore del campo di testo `inField` nel documento HTML. Questa funzione viene chiamata dal documento Flash nel momento in cui l'utente fa clic sul componente Button `send_button`.

- Aggiungere il codice HTML seguente prima di chiudere il tag `</body>`:

```

<form name="outForm" method="POST"
 action="javascript:makeCall(document.outForm.outField.value);">
 Sending to AS:

 <input type="text" name="outField" value="" />

 <input type="submit" value="Send" />
</form>

<form name="inForm" method="POST" action="">
 Receiving from AS:

 <input type="text" name="inField">
</form>
```

Questo codice HTML crea due moduli HTML simili ai moduli creati in ambiente Flash nell'esercizio precedente. Il primo modulo invia il valore del campo di testo `outField` alla funzione di JavaScript `makeCall()` definita in una fase precedente. Il secondo modulo si utilizza per visualizzare un valore che viene inviato dal file SWF quando l'utente fa clic sull'istanza `send_button`.

- Salvare il documento HTML e caricare i file HTML e SWF su un server Web.
- Visualizzare il file HTML in un browser Web, inserire una stringa nell'istanza `TextInput out_ti`, quindi fare clic sul pulsante Invia.  
Flash chiama la funzione JavaScript `jsFunc()` e passa il contenuto del campo di testo `out_ti`, che visualizza il contenuto del campo di testo `inField` del modulo HTML `inField`.
- Digitare un valore nel campo di testo HTML `outField`, quindi fare clic sul pulsante Invia.  
Flash chiama la funzione del file SWF `asFunc()` che visualizza la stringa nell'istanza `TextInput in_ti`.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, ExtInt.fla.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI\simple example*.
- In Macintosh, scegliere *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/ExternalAPI/simple example*.

Per un esempio dimostrativo completo che utilizza l'API External, consultare “[Controllo di Flash Video con l'API External](#)” a pagina 727. Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

**NOTA**

Evitare l'uso di altri metodi di accesso all'oggetto plug-in, come `document.getElementById("pluginName")` o `document.all.pluginName`, perché tali metodi non funzionano correttamente in tutti i browser.

## Controllo di Flash Video con l'API External

La procedura seguente mostra come controllare i file FLV (Flash Video) mediante comandi in una pagina HTML e visualizza le informazioni relative al video in un campo di testo HTML. Questa procedura utilizza l'API External per eseguire la funzionalità.

### Per costruire un'applicazione Flash con l'API External:

1. Creare un nuovo documento Flash e salvarlo come **video.fla**.
2. Aggiungere un nuovo simbolo video alla libreria selezionando Nuovo video dal menu a comparsa nel pannello Libreria.
3. Trascinare un simbolo video nello stage e assegnare ad esso il nome di istanza **selected\_video**.
4. Selezionare l'istanza **selected\_video**, quindi la finestra di ispezione Proprietà per ridimensionare l'istanza a 320 pixel per larghezza e 240 pixel per altezza.
5. Impostare le coordinate **x** e **y** per la posizione del video a 0.
6. Selezionare lo stage ed utilizzare la finestra di ispezione Proprietà per ridimensionarne le dimensioni a 320 pixel per 240 pixel.

Ora lo stage corrisponde alle dimensioni dell'istanza video.

**7. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:**

```
import flash.external.ExternalInterface;

/* Registrare playVideo() e pauseResume() in modo da permettere
a JavaScript di chiamarli nella pagina contenitore di HTML.*/
ExternalInterface.addCallback("playVideo", null, playVideo);
ExternalInterface.addCallback("pauseResume", null, pauseResume);

/* Il video richiede un oggetto NetConnection e NetStream.*/
var server_nc:NetConnection = new NetConnection();
server_nc.connect(null);
var video_ns:NetStream = new NetStream(server_nc);

/* Collegare l'oggetto NetStream object all'oggetto Video object nello
stage in modo da
visualizzare i dati NetStream nell'oggetto Video.*/
selected_video.attachVideo(video_ns);

/* Il metodo onStatus() viene chiamato automaticamente quando lo stato
dell'oggetto NetStream viene aggiornato (ad esempio, inizia la
riproduzione del video).
Quando si verifica, inviare il valore della proprietà del codice alla
pagina HTML
chiamando la funzione di JavaScript updateStatus() per mezzo di
ExternalInterface.*/
video_ns.onStatus = function(obj:Object):Void {
 ExternalInterface.call("updateStatus", " " + obj.code);
};

function playVideo(url:String):Void {
 video_ns.play(url);
}

function pauseResume():Void {
 video_ns.pause();
}
```

La prima parte di questo codice ActionScript definisce due funzioni ExternalInterface di callback: functions, playVideo() e pauseResume(). Tali funzioni vengono chiamate da JavaScript nella procedura successiva. La seconda parte del codice crea un nuovo oggetto NetConnection e NetStream che si utilizza con l'istanza video per riprodurre in modo dinamico i file FLV.

Il codice nella procedura successiva definisce un gestore di eventi `onStatus` per l'oggetto `NetStream video_ns`. Quando l'oggetto `NetStream` modifica il proprio stato, Flash utilizza il metodo `ExternalInterface.call()` per attivare la funzione JavaScript, `updateStatus()`. Le due funzioni finali, `playVideo()` e `pauseResume()`, controllano la riproduzione dell'istanza video nello stage. Entrambe vengono chiamate da JavaScript riportato nella procedura seguente.

8. Salvare il documento Flash.
9. Selezionare File > Impostazioni pubblicazione, quindi la scheda Formati ed assicurarsi che sia selezionato sia HTML sia Flash.
10. Fare clic su Pubblica per pubblicare i file SWF e HTML sul disco rigido.

Al termine, passare alla procedura successiva per creare il contenitore del file SWF.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, `external.fla`.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples\ActionScript/ExternalAPI*.

Nella procedura seguente modificare il codice HTML generato da Flash nella precedente procedura. Tale procedura crea il codice JavaScript e HTML necessario per consentire la riproduzione di file FLV nel file SWF.

#### Per creare il contenitore per il file SWF:

1. Completare la procedura precedente.
2. Aprire il documento `video.html` pubblicato nell'ultima fase della procedura precedente.
3. Modificare il codice esistente affinché corrisponda a quello seguente:

**NOTA** Esaminare i commenti al codice nell'esempio seguente: una panoramica del codice segue l'esempio del codice.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>ExternalInterface</title>

<script language="JavaScript">
 // Utilizzare una variabile per fare riferimento al file SWF
 incorporato.
```

```

var flashVideoPlayer;

/* Quando si carica la pagina HTML, attraverso l'evento onLoad del tag
<body>, questa chiama la funzione initialize(). */
function initialize() {
 /* Verificare che il browser Web sia IE. In caso affermativo
 flashVideoPlayer è
 window.videoPlayer. In caso negativo, è document.videoPlayer. Il
 videoPalyer corrisponde all'ID assegnato ai tag <object> ed <embed>. */
 var isIE = navigator.appName.indexOf("Microsoft") != -1;
 flashVideoPlayer = (isIE) ? window['videoPlayer'] :
 document['videoPlayer'];
}

/* Quando l'utente fa clic sul pulsante per riproduzione nel modulo,
aggiornare l'area di testo videoStatus e chiamare la funzione
playVideo() nel file SWF file, passandolo all'URL del file FLV. */
function callFlashPlayVideo() {
 var comboBox = document.forms['videoForm'].videos;
 var video = comboBox.options[comboBox.selectedIndex].value;
 updateStatus("____" + video + "____");
 flashVideoPlayer.playVideo("http://www.helpexamples.com/flash/
video/" + video);
}

// Chiamare la funzione pauseResume() nel file SWF.
function callFlashPlayPauseVideo() {
 flashVideoPlayer.pauseResume();
}

/* La funzione updateStatus() viene chiamata dal file SWF file dal
metodo onStatus() dell'oggetto NetStream. */
function updateStatus(message) {
 document.forms['videoForm'].videoStatus.value += message + "\n";
}
</script>
</head>
<body bgcolor="#ffffff" onLoad="initialize();">

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
swflash.cab#version=8,0,0,0" width="320" height="240" id="videoPlayer"
align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="video.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />

```

```

<embed src="video.swf" quality="high" bgcolor="#ffffff" width="320"
 height="240" name="videoPlayer" align="middle"
 allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
 pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>

<form name="videoForm">
 Select a video:

 <select name="videos">
 <option value="lights_long.flv">lights_long.flv</option>
 <option value="clouds.flv">clouds.flv</option>
 <option value="typing_long.flv">typing_long.flv</option>
 <option value="water.flv">water.flv</option>
 </select>
 <input type="button" name="selectVideo" value="play"
 onClick="callFlashPlayVideo();" />

 Playback <input type="button" name="playPause" value="play/pause"
 onClick="callFlashPlayPauseVideo();" />

 Video status messages

 <textarea name="videoStatus" cols="50" rows="10"></textarea>
</form>

</body>
</html>

```

Questo codice HTML definisce quattro funzioni JavaScript: `initialize()`, `callFlashPlayVideo()`, `callFlashPlayPauseVideo()` e `updateStatus()`. La funzione `initialize()` viene chiamata all'interno del tag `body` nell'evento `onLoad`. Entrambe le funzioni `callFlashPlayVideo()` e `callFlashPlayPauseVideo()` vengono chiamate quando l'utente fa clic sul pulsante di riproduzione o riproduzione/pausa nel documento HTML ed attiva le funzioni `playVideo()` e `pauseResume()` nel file SWF. La funzione finale, `updateStatus()`, viene chiamata dal file SWF quando si attiva il gestore di eventi `onStatus` dell'oggetto `NetStream video_ns`. Questo codice HTML definisce anche un modulo con una casella combinata di video in cui l'utente può eseguire delle selezioni. Quando l'utente seleziona un video e fa clic sul pulsante di riproduzione, viene chiamata la funzione di JavaScript `callFlashPlayVideo()` che a sua volta chiama la funzione `playVideo()` all'interno del file SWF. Questa funzione passa l'URL del file SWF per il caricamento nell'istanza video. Durante la riproduzione del video e la modifica dello stato dell'oggetto `NetStream`, viene aggiornato il contenuto dell'area di testo HTML sullo stage.

4. Salvare le modifiche del documento HTML, quindi caricare i file HTML e SWF su un sito Web.
5. Aprire il documento remoto video.html dal sito Web, selezionare un video dalla casella combinata, quindi fare clic sul pulsante di riproduzione.  
Flash riproduce il file FLV selezionato e aggiorna il contenuto dell'area di testo videoStatus nel documento HTML.

Nella cartella Samples presente sul disco rigido è possibile trovare il file sorgente di esempio, external.fla.

- In Windows, accedere a *unità di avvio\Programmi\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI*.
- In Macintosh, accedere a *Macintosh HD/Applicazioni/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/ExternalAPI*.

Per ulteriori informazioni sull'API External, consultare `ExternalInterface` (`flash.external.ExternalInterface`) nella *Guida di riferimento di ActionScript 2.0*.

Per ulteriori informazioni sulla sicurezza dei file locali, consultare “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

**NOTA**

Evitare l'uso di altri metodi di accesso all'oggetto plug-in, come `document.getElementByld("pluginName")` o `document.all.pluginName`, perché tali metodi non funzionano correttamente in tutti i browser.

# Nozioni fondamentali sulla sicurezza

17

In Macromedia Flash Basic 8 e Macromedia Flash Professional 8, il codice ActionScript può essere utilizzato per caricare dati da origini esterne in un file SWF o inviare i dati a un server. Quando si caricano i dati in un file SWF, è necessario comprendere e adattare il modello di sicurezza di Flash 8. Quando si apre un file SWF sul disco rigido, potrebbe essere necessario eseguire configurazioni speciali in modo da poter provare il file localmente.

Per informazioni sulla sicurezza dei file locali, vedere “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735. Per informazioni sulle differenze tra il modello di sicurezza di Flash Player 7 e Flash Player 8, vedere “[Informazioni sulla compatibilità con i modelli di sicurezza di Flash Player precedenti](#)” a pagina 734. Per informazioni su come caricare e analizzare i dati da un server, leggere il [Capitolo 16, “Operazioni con i dati esterni”](#) a pagina 689. Per ulteriori informazioni sulla sicurezza, vedere [www.macromedia.com/devnet/security](http://www.macromedia.com/devnet/security) e [www.macromedia.com/software/flashplayer/security/](http://www.macromedia.com/software/flashplayer/security/).

Per ulteriori informazioni sulla sicurezza in Flash 8, consultare i seguenti argomenti:

<b>Informazioni sulla compatibilità con i modelli di sicurezza di Flash Player precedenti</b> .....	734
<b>Informazioni sulla sicurezza dei file locali e Flash Player</b> .....	735
<b>Informazioni su domini, sicurezza tra domini e file SWF</b> .....	753
<b>File di criteri server-side per l'accesso ai dati</b> .....	761
<b>Accesso tra file SWF da HTTP a HTTPS</b> .....	767

# Informazioni sulla compatibilità con i modelli di sicurezza di Flash Player precedenti

In seguito alle modifiche apportate alle funzionalità di sicurezza di Flash Player 7, è possibile che il contenuto che viene eseguito correttamente in Flash Player 6 o nelle versioni precedenti non funzioni correttamente nelle versioni successive di Flash Player. Con Flash Player 6, ad esempio, un file SWF presente all'indirizzo www.macromedia.com poteva leggere i dati su un server all'indirizzo data.macromedia.com; in altri termini, Flash Player 6 consentiva a un file SWF, appartenente a un determinato dominio, di caricare dati da un dominio simile.

Con Flash Player 7 e le versioni successive, se un file SWF della versione 6 o di versioni precedenti tenta di caricare dati da un server di un dominio diverso e questo server non fornisce un file di criteri che consente la lettura dal dominio del file SWF, viene visualizzata la finestra di dialogo Impostazioni Flash Player che chiede all'utente di consentire o negare l'accesso ai dati appartenenti a un dominio diverso.

Se l'utente consente l'accesso, il file SWF può accedere ai dati richiesti. In caso contrario, l'accesso ai dati da parte del file SWF non è possibile.

Per evitare la visualizzazione della finestra di dialogo, creare un file di criteri sul server che fornisce i dati. Per ulteriori informazioni, vedere [“Come consentire il caricamento di dati tra domini diversi” a pagina 762](#).

Flash Player 7 e le versioni successive non consentono l'accesso a più domini senza un file dei criteri di sicurezza.

Flash Player 8 ha modificato il modo di gestire System.security.allowDomain. Un file SWF di Flash 8 che chiama System.security.allowDomain con qualunque argomento, o qualsiasi altro file SWF che utilizzi il valore carattere jolly (\*), consente l'accesso solo a se stesso. È disponibile ora il supporto per un valore carattere jolly (\*), ad esempio:

System.security.allowDomain("\*") e System.security.allowInsecureDomain("\*").

Se un file SWF della versione 7 o di una versione precedente chiama System.security.allowDomain o System.security.allowInsecureDomain con un argomento diverso dal carattere jolly (\*), questo interesserà tutti i file SWF della versione 7 o di una versione precedente nella chiamata del dominio del file SWF, come accadeva in Flash Player 7. Tuttavia, questo tipo di chiamata non incide sui file SWF di Flash Player 8 (o versioni successive) nella chiamata del dominio del file SWF. In questo modo è possibile ridurre al minimo il mancato utilizzo del contenuto precedente in Flash Player.

Per ulteriori informazioni, vedere “[Informazioni su domini, sicurezza tra domini e file SWF](#)” a pagina 753, `allowDomain` (`security.allowDomain` method), e `allowInsecureDomain` (`security.allowInsecureDomain` method).

Flash Player 8 non consente ai file SWF locali di comunicare con Internet senza una configurazione specifica sul computer. Si supponga che prima dell'applicazione di queste restrizioni sia stato pubblicato contenuto. Se questo contenuto tenta di comunicare con la rete o il file system locale, o con entrambi, Flash Player 8 interrompe il funzionamento e l'utente deve fornire esplicitamente l'autorizzazione per garantire il corretto funzionamento dell'applicazione. Per ulteriori informazioni, vedere “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735

## Informazioni sulla sicurezza dei file locali e Flash Player

Flash Player 8 ha apportato miglioramenti al modello di sicurezza, in cui alle applicazioni Flash e ai file SWF su un computer locale non è consentito di comunicare *sia* con Internet che con il file system locale per impostazione predefinita. Per *file SWF locale* si intende un file SWF installato localmente sul computer di un utente e non servito da un sito Web, che non include file di proiettore (EXE).

**NOTA**

Le restrizioni descritte in questa sezione non incidono sui file SWF che si trovano in Internet.

Quando si crea un file FLA, è possibile indicare se a un file SWF è consentito comunicare con una rete o con un file system locale. Nelle versioni precedenti di Flash Player, i file SWF locali potevano interagire con altri file SWF e caricare i dati da qualsiasi postazione remota o locale. In Flash Player 8, un file SWF non può eseguire connessioni al file system locale *e* a Internet. In questo modo si aumenta la sicurezza: un file SWF non può leggere i file presenti sul disco rigido e inviare i contenuti di tali file attraverso Internet.

Questa restrizione di sicurezza influisce solo sul contenuto distribuito localmente, che sia contenuto precedente (un file FLA creato con un versione antecedente di Flash) o creato in Flash 8. Se, ad esempio, si distribuisce un'applicazione Flash tramite Flash MX 2004 o una versione precedente e questa applicazione viene eseguita localmente e accede anche a Internet, in Flash Player 8 viene chiesto all'utente l'autorizzazione esplicita per la comunicazione con Internet.

Quando si verifica un file presente sul disco rigido, esistono diversi passaggi per stabilire se il file è un documento locale affidabile o un documento potenzialmente pericoloso. Se si crea il file nell'ambiente di creazione di Flash (per esempio, quando si seleziona Ctrl > Prova filmato), il file è affidabile perché si trova in un ambiente di prova.

In Flash Player 7 e nelle versioni precedenti, i file SWF locali disponevano di autorizzazioni che consentivano di leggere sia dal file system locale che dalla rete (ad esempio Internet). In Flash Player 8, i file SWF locali possono avere i seguenti livelli di autorizzazioni:

**Accesso solo al file system locale (predefinito)** Un file SWF locale può leggere dal file system locale e dai percorsi di rete UNC (Universal Naming Convention) e non può comunicare con Internet. Per ulteriori informazioni sull'accesso dei file SWF ai file locali, vedere “[Accedi solo ai file locali \(livello predefinito\)](#)” a pagina 744.

**Accesso solo alla rete** Un file SWF locale può accedere alla rete (ad esempio Internet) ma non al file system locale in cui è installato. Per ulteriori informazioni sull'accesso dei file SWF solo alla rete, vedere “[Accedi solo alla rete](#)” a pagina 745.

**Accesso alla rete e al file system locale** Un file SWF locale può leggere dal file system locale in cui è installato, leggere e scrivere sui server e può eseguire script su altri file SWF presenti in rete o nel file system locale. Questi file sono affidabili e il loro comportamento è analogo a quello che avevano in Flash Player 7. Per ulteriori informazioni sull'accesso locale e di rete dei file SWF, vedere “[Accesso alla rete e al file system](#)” a pagina 745.

Per ulteriori informazioni sulla sicurezza dei file locali in Flash 8, poiché attiene allo strumento di creazione, consultare le sezioni seguenti:

- “[Nozioni fondamentali sulle funzioni di sicurezza sandbox locali](#)” a pagina 737
- “[Informazioni sulle impostazioni di sicurezza di Flash Player](#)” a pagina 738
- “[Informazioni sulla sicurezza dei file locali e i file di proiettori](#)” a pagina 740
- “[Informazioni sulla risoluzione dei problemi relativi a file SWF preesistenti](#)” a pagina 741
- “[Risoluzione dei problemi relativi al contenuto precedente distribuito su computer locali](#)” a pagina 742
- “[Pubblicazione dei file per la distribuzione locale](#)” a pagina 743

Per informazioni sulla sicurezza dei file locali per gli utenti, vedere “[Informazioni sulle impostazioni di sicurezza di Flash Player](#)” a pagina 738. Per ulteriori informazioni sulla sicurezza, vedere [www.macromedia.com/devnet/security/](http://www.macromedia.com/devnet/security/) e [www.macromedia.com/software/flashplayer/security/](http://www.macromedia.com/software/flashplayer/security/).

## Nozioni fondamentali sulle funzioni di sicurezza sandbox locali

In Flash Player sono disponibili diverse funzioni di sicurezza sandbox. Ciascuna di esse determina la modalità di interazione di un file SWF con il file system locale, con la rete, o sia con il file system locale che con la rete contemporaneamente. L'esistenza di restrizioni sulla modalità di interazione di un file con il file system locale o con la rete garantisce un maggiore livello di sicurezza del computer e dei file. La conoscenza delle nozioni fondamentali delle funzioni di sicurezza sandbox consente di sviluppare e verificare le applicazioni Flash sul computer senza incorrere in errori imprevisti.

### Local-with-file-system

Per aumentare la sicurezza, Flash Player 8 colloca tutti i file SWF locali, inclusi tutti i file SWF locali precedenti, nella funzione di sicurezza sandbox local-with-file-system, per impostazione predefinita (a meno che non venga eseguita un'impostazione diversa). Per alcuni file SWF precedenti (versione precedente a Flash Player 8), le operazioni potrebbero essere influenzate dall'applicazione di limitazioni sul loro accesso (nessun accesso esterno alla rete), ma questo fornisce l'impostazione predefinita più sicura per la protezione degli utenti.

Da questa funzione di sicurezza sandbox, i file SWF possono leggere dai file sui file system locali o da percorsi di rete UNC (utilizzando ad esempio il metodo `XML.load()`), ma non possono comunicare con la rete. Ciò garantisce all'utente che i dati locali non possano essere diffusi nella rete o altrimenti condivisi in modo inadeguato.

### Local-with-networking

Quando i file SWF locali vengono assegnati alla funzione di protezione sandbox local-with-networking, rinunciano all'accesso al file system locale. In compenso, ai file SWF è consentito l'accesso alla rete. Tuttavia, a un file SWF di tipo local-with-networking non è ancora consentito leggere dati ottenuti dalla rete a meno che non siano presenti autorizzazioni per quella determinata azione. Pertanto, un file SWF local-with-networking non ha l'accesso locale, ma è in grado di trasmettere i dati sulla rete e può leggere i dati di rete da quei siti che definiscono autorizzazioni di accesso specifiche per il sito.

### Local-trusted

I file SWF assegnati alla funzione di sicurezza local-trusted possono interagire con qualunque altro file SWF e caricare i dati da qualsiasi postazione (remota o locale).

## Informazioni sulle impostazioni di sicurezza di Flash Player

Macromedia ha progettato Flash Player in modo da fornire impostazioni di sicurezza che non richiedono all'utente di autorizzare o negare in modo esplicito l'accesso nella maggior parte dei casi. Talvolta ci si può imbattere in contenuto di Flash precedente che era stato creato utilizzando regole di sicurezza precedenti per Flash Player 7 o versione precedente. In tali casi, Flash Player consente l'uso del contenuto nel modo previsto dallo sviluppatore, utilizzando le regole di sicurezza precedenti; oppure, è possibile scegliere di applicare le regole aggiornate e più rigorose. Quest'ultima alternativa garantisce solo la visualizzazione o riproduzione del contenuto che soddisfa gli standard di sicurezza più recenti, ma talvolta può impedire il corretto funzionamento del contenuto Flash precedente.

Gli utenti che visualizzano i file SWF (inclusi gli sviluppatori non Flash) possono impostare autorizzazioni a livello globale tramite il pannello Impostazioni globali di sicurezza in Gestione impostazioni di Flash Player (indicato nella figura seguente).



Quando il contenuto precedente viene eseguito in una nuova versione del player e Flash Player ha bisogno che l'utente decida se applicare o meno le nuove regole, è possibile che venga visualizzata una delle seguenti finestre di dialogo. Tali finestre di dialogo richiedono l'autorizzazione prima di consentire al contenuto Flash precedente di comunicare con altre posizioni in Internet:

- Potrebbe essere visualizzata una finestra di dialogo che avvisa l'utente che il contenuto Flash che viene utilizzato sta tentando di usare le regole di sicurezza precedenti per accedere alle informazioni da un sito esterno al proprio dominio e che le informazioni potrebbero essere condivise tra i due siti. Flash Player chiede se si desidera consentire o negare l'accesso.

Oltre a rispondere alla finestra di dialogo, è possibile usare il pannello Impostazioni globali di sicurezza per specificare se Flash Player debba richiedere sempre l'autorizzazione, tramite la finestra di dialogo, prima di consentire l'accesso; negare sempre l'accesso, senza chiedere prima; o consentire sempre l'accesso ad altri siti o domini senza richiesta di autorizzazione.

- Potrebbe essere visualizzata una finestra di dialogo che avvisa l'utente che un file SWF sta tentando di comunicare con Internet. (Solo Flash Player 8). Per impostazione predefinita Flash Player 8 non consente al contenuto Flash locale di comunicare con Internet.



Fare clic su Impostazioni per accedere al pannello Impostazioni globali di sicurezza, in cui è possibile specificare che talune applicazioni Flash presenti sul computer possono comunicare con Internet.

Per modificare le impostazioni sulla sicurezza o per informazioni sulle opzioni, viene utilizzato il pannello Impostazioni globali di sicurezza. Usare questo pannello per ripristinare le impostazioni di riservatezza in Macromedia Flash Player:

- Se si seleziona *Nega sempre* e si conferma la selezione, viene negato l'accesso a qualsiasi sito Web che tenta di utilizzare la videocamera o il microfono. All'utente non viene chiesto di nuovo se un sito Web può utilizzare la videocamera o il microfono. Questa impostazione vale sia per i siti Web che l'utente ha già visitato che per quelli non ancora visitati.
- Se si seleziona *Chiedi sempre* e si conferma la selezione, qualsiasi sito Web che tenta di utilizzare la videocamera o il microfono, deve richiedere l'autorizzazione. Questa impostazione vale sia per i siti Web che l'utente ha già visitato che per quelli non ancora visitati.

Se in precedenza è stato selezionato Ricorda nel pannello delle impostazioni della riservatezza (come nella figura seguente) per autorizzare o negare in modo permanente l'accesso per uno o più siti Web, la selezione di Chiedi sempre o Nega sempre ha l'effetto di deselectare Ricorda per tutti quei siti Web. In altre parole, la selezione effettuata in questa sede sostituisce tutte le selezioni precedenti che siano state eseguite nel pannello delle impostazioni della riservatezza, come nella figura seguente.



Dopo aver selezionato Chiedi sempre o Nega sempre (o invece di fare così), è possibile specificare le impostazioni di riservatezza dei singoli siti Web che sono già stati visitati. Ad esempio, è possibile selezionare qui Nega sempre, quindi utilizzare il pannello Impostazioni sulla riservatezza dei siti Web e selezionare Consenti sempre per singoli siti Web conosciuti e affidabili.

Per il contenuto distribuito a livello locale e i dati locali è presente un'altra opzione: è possibile specificare quali file SWF possono accedere a Internet utilizzando il pannello Impostazioni globali di sicurezza. Per ulteriori informazioni sulla specifica delle impostazioni nel pannello Impostazioni globali di sicurezza, vedere ["Specifiche dei file affidabili utilizzando Gestione impostazioni" a pagina 746](#). Per ulteriori informazioni sul pannello Impostazioni globali di sicurezza, vedere [www.macromedia.com/support/documentation/en/flashplayer/help/settings\\_manager04a.html](http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04a.html).



Le impostazioni del pannello Impostazioni globali di sicurezza hanno la precedenza su tutte le altre selezioni effettuate nella finestra di dialogo a comparsa sulla sicurezza.

## Informazioni sulla sicurezza dei file locali e i file di proiettori

I file di proiettori e i file SWF che vi sono contenuti o caricati nel proiettore in fase di runtime non vengono interessati dalle restrizioni di sicurezza per i file locali, poiché l'utente finale deve avviare l'eseguibile per utilizzare il file SWF. Non vi sono modifiche alla sicurezza e ai file di proiettori in Flash Player 8 che dispone dello stesso livello di accesso e di sicurezza delle versioni precedenti di Flash Player.

Gli utenti sono spesso cauti nell'esecuzione dei file di proiettori. Un file di proiettore è un'applicazione EXE o Macintosh eseguibile e gli utenti dovrebbero prestare attenzione all'esecuzione di tali file sui loro computer. Se viene distribuita un'applicazione mediante i file di proiettore, alcuni utenti potrebbero non installarla.

Inoltre un file di proiettore incorpora una versione specifica di Flash Player all'interno del proiettore, che potrebbe essere precedente all'ultima versione di Flash Player scaricabile dal sito Web di Macromedia. Flash Player incorporato con il file di proiettore potrebbe essere una versione non aggiornata se il proiettore è stato creato con una versione precedente di Flash, oppure un'edizione di Flash Player è stata rilasciata dopo la versione corrente dello strumento di creazione di codice di Flash. Per tali motivi, è necessario distribuire le applicazioni utilizzando i file SWF, laddove sia possibile.

## Informazioni sulla risoluzione dei problemi relativi a file SWF preesistenti

Alcuni file FLA e SWF preesistenti (creati con Flash MX 2004 e versioni precedenti) potrebbero non funzionare quando vengono provati o distribuiti a livello locale (su un disco rigido) a causa delle modifiche relative alla sicurezza in Flash 8. Questo potrebbe verificarsi quando un file SWF tenta di accedere ai siti Web al di fuori del proprio dominio, e, in questo caso, è necessario implementare un file di criteri validi tra più domini.

Ci potrebbero essere file FLA o SWF creati in Flash MX 2004 o versione precedente che sono stati distribuiti ad utenti che non utilizzano lo strumento di creazione di codice di Flash 8 ma che hanno effettuato l'aggiornamento a Flash Player 8. Se il contenuto distribuito o provato a livello locale in precedenza, ad esempio un file SWF già presente sul disco rigido dell'utente, non può essere utilizzato perché tenta di comunicare con Internet quando viene riprodotto in Flash Player 8, è necessario che gli utenti considerino il contenuto affidabile in modo esplicito per garantirne la corretta riproduzione (facendo clic su un pulsante in una finestra di dialogo).

Per informazioni su come risolvere i problemi relativi al contenuto precedente per la riproduzione su un computer locale, vedere “[Risoluzione dei problemi relativi al contenuto precedente distribuito su computer locali](#)” a pagina 742.

## Risoluzione dei problemi relativi al contenuto precedente distribuito su computer locali

Se i file SWF pubblicati per Flash Player 7 o versioni precedenti sono distribuiti su computer locali e comunicano anche con Internet, è necessario che gli utenti consentano la comunicazione con Internet in modo esplicito. Gli utenti possono consentire l'accesso al contenuto aggiungendo il percorso del file SWF sul proprio computer locale alla funzione di sicurezza sandbox trusted in Gestione impostazioni.

**Per impostare i file SWF per la riproduzione locale, utilizzare una delle seguenti opzioni:**

**Ridistribuisci** Eseguire l'utilità Local Content Updater. In questo modo il file SWF viene riconfigurato per essere compatibile con il modello di sicurezza di Flash Player 8 e per accedere solo alla rete o solo al file system locale. Per ulteriori informazioni e per scaricare l'utilità Local Content Updater, vedere [www.macromedia.com/support/flashplayer/downloads.html](http://www.macromedia.com/support/flashplayer/downloads.html).

**Ripubblica e ridistribuisci** Ripubblicare il file con Flash Basic 8 o Flash Professional 8. Lo strumento di creazione di codice richiede di specificare nella finestra di dialogo Impostazioni pubblicazione se un file SWF locale possa accedere alla rete o al file system locale, ma non a entrambi. Se si consente a un file SWF locale di accedere alla rete, è necessario anche autorizzare quel file SWF (e tutti i file SWF locali) nei file SWF, nei file HTML, nei file di dati e/o nei server a cui il file accede. Per ulteriori informazioni, vedere “[Pubblicazione dei file per la distribuzione locale](#)” a pagina 743.

**Distribuisci nuovo contenuto** Utilizzare un file di configurazione (.cfg) nella cartella #Security/FlashPlayerTrust. È possibile utilizzare questo file per impostare autorizzazioni di accesso di rete e locale. Per ulteriori informazioni, vedere “[Creazione di file di configurazione per lo sviluppo di Flash](#)” a pagina 748.

**NOTA**

Tutte e tre le opzioni richiedono la ripubblicazione o la ridistribuzione del file SWF.

## Pubblicazione dei file per la distribuzione locale

È possibile inviare i file FLA o SWF di Flash 8 ad un utente perché li provi o li approvi ed è necessario che l'applicazione acceda a Internet. Se il documento viene riprodotto su un sistema locale ma accede ai file in Internet (ad esempio, caricando codice XML o inviando variabili), potrebbe essere necessario un file di configurazione per garantire il corretto funzionamento del contenuto, o potrebbe essere necessario impostare il file FLA in modo tale che il file SWF che viene pubblicato, possa accedere alla rete. In alternativa, è possibile impostare un file di configurazione nella directory FlashPlayerTrust. Per ulteriori informazioni sull'impostazione dei file di configurazione, vedere “[Creazione di file di configurazione per lo sviluppo di Flash](#)” a pagina 748.

Utilizzare Flash Basic 8 o Flash Professional 8 per creare contenuto per la distribuzione locale che funzioni con il modello di sicurezza dei file locali di Flash Player 8. In Impostazioni pubblicazione di Flash 8 è necessario specificare se il contenuto locale può accedere alla rete o al file system locale. Non è possibile accedere a entrambi.

È possibile impostare livelli di autorizzazioni per un file FLA nella finestra di dialogo Impostazioni pubblicazione. Tali livelli di autorizzazioni incidono sulla riproduzione locale del file FLA, quando viene eseguita a livello locale sul disco rigido.



Se si consente a un file locale di accedere alla rete, è necessario anche autorizzare quel file SWF nei file SWF, nei file HTML, nei file di dati e nei server a cui il file accede.

**File SWF in rete** I file SWF che vengono scaricati da una rete (ad esempio un server in linea) sono inseriti in una *funzione di sicurezza sandbox* separata che corrisponde ai loro domini di origine univoci del sito Web. I file SWF locali che specificano che sono dotati di accesso alla rete sono inseriti nella funzione di sicurezza sandbox *local-with-networking*. Per impostazione predefinita, questi file possono leggere i dati solo dallo stesso sito di origine. Il criterio di corrispondenza esatta dei domini si applica a questi file. I file SWF in rete possono accedere ai dati provenienti da altri domini se dispongono delle autorizzazioni appropriate. Per ulteriori informazioni sui file SWF in rete, vedere “[Accedi solo alla rete](#)” a pagina 745.

**File SWF locali** I file SWF che utilizzano i file system locali o percorsi di rete UNC sono inseriti in una delle tre funzioni di sicurezza sandbox in Flash Player 8. Per impostazione predefinita, i file SWF locali sono inseriti nella funzione di sicurezza sandbox *local-with-file-system*. I file SWF locali registrati come affidabili (tramite un file di configurazione) sono inseriti nella funzione di sicurezza sandbox *local-trusted*. Per informazioni sulle tre funzioni di sicurezza sandbox, vedere “[Accedi solo ai file locali \(livello predefinito\)](#)” a pagina 744.

Per ulteriori informazioni sulla funzione di sicurezza sandbox, vedere “[Nozioni fondamentali sulle funzioni di sicurezza sandbox locali](#)” a pagina 737.

I primi due livelli di autorizzazioni vengono impostati nell'ambiente di creazione di Flash e il terzo viene impostato utilizzando il pannello Impostazioni globali di sicurezza o il file FlashAuthor.cfg. L'esempio seguente mostra le opzioni disponibili quando si pubblica un file per la prova sul disco rigido.

#### **Per pubblicare un documento con un livello di autorizzazioni specificato:**

1. Aprire il file FLA per il quale si desidera specificare un livello di autorizzazioni.
2. Scegliere File > Impostazioni pubblicazione > Flash.
3. Trovare la finestra di dialogo Sicurezza riproduzione locale e scegliere una delle opzioni seguenti dal menu a comparsa:
  - Accedi solo ai file locali (Vedere “[Accedi solo ai file locali \(livello predefinito\)](#)”)
  - Accedi solo alla rete (Vedere “[Accedi solo alla rete](#)”)
4. Fare clic su OK per continuare la creazione del file FLA, o fare clic su Pubblica per creare il file SWF.

Per ulteriori informazioni sui livelli di autorizzazioni che è possibile impostare per le applicazioni, vedere “[Accedi solo ai file locali \(livello predefinito\)](#)” a pagina 744, “[Accedi solo alla rete](#)” a pagina 745 e “[Accesso alla rete e al file system](#)” a pagina 745.

#### **Accedi solo ai file locali (livello predefinito)**

Per impostare questo livello di autorizzazioni, selezionare Impostazioni pubblicazione > Flash e quindi scegliere Accedi solo ai file locali dal menu a comparsa Sicurezza riproduzione locale. Questo livello di autorizzazioni consente a un file SWF locale di accedere solo al file system locale in cui viene eseguito il file SWF. Il file SWF può leggere dati da file noti sul disco locale senza alcuna restrizione. Tuttavia, si applicano le seguenti restrizioni all'applicazione che accede alla rete:

- Il file SWF non può accedere alla rete, né eseguire script su file SWF in rete. I file SWF in rete non possono eseguire script su questo file.
- Il file SWF non può comunicare con file SWF locali con autorizzazione di accesso solo alla rete, né con pagine HTML. Tuttavia, in alcuni casi la comunicazione è consentita, ad esempio se la pagina HTML è affidabile e `allowScriptAccess` è impostato su `always` o se `allowScriptAccess` non è impostato e il file SWF è per Flash Player 7 o versione precedente.

## Accedi solo alla rete

Per impostare questo livello di autorizzazioni, selezionare Impostazioni pubblicazione > Flash e quindi scegliere Accedi solo alla rete dal menu a comparsa Sicurezza riproduzione locale. I file SWF locali con accesso alla rete possono leggere dati da un server, se il server contiene un file di criteri dei domini con l'istruzione `<allow-access-from-domain= “*”>`. Possono eseguire script su altri file SWF se questi file, a cui si accede, contengono `System.security.allowDomain(“*”)`. Su un file SWF locale con accesso alla rete è possibile inoltre eseguire script da file SWF in rete se il file locale contiene `allowDomain(“*”)`. Il file SWF non può leggere dati da file locali. In alcuni casi, il tipo di file SWF influenza sull'accesso. Per informazioni, vedere `allowDomain (security.allowDomain method)` nella *Guida di riferimento di ActionScript 2.0*.

Il valore carattere jolly (\*) indica che l'accesso è consentito a *tutti* i domini, inclusi gli host locali. Utilizzare l'argomento carattere jolly solo se si desidera garantire un accesso così ampio. Senza queste autorizzazioni, i file SWF locali con accesso alla rete possono comunicare solo con altri file SWF locali che dispongono di accesso alla rete e possono inviare dati ai server (utilizzando, ad esempio, `XML.send()`). In alcuni casi, l'accesso è consentito se il file HTML è affidabile.

## Accesso alla rete e al file system

Si tratta del livello di autorizzazioni maggiore. Un file SWF locale che dispone di queste autorizzazioni viene definito file SWF *locale affidabile*. I file SWF locali affidabili possono leggere da altri file SWF locali, interagire con tutti i server e scrivere codice ActionScript per altri file SWF o file HTML ai quali non è stata vietata esplicitamente l'autorizzazione ai file (ad esempio, con `allowScriptAccess="none"`). Questo livello di autorizzazione può essere concesso dall'utente o dallo sviluppatore Flash nei seguenti modi:

- Utilizzando il pannello Impostazioni globali di sicurezza in Gestione impostazioni.
- Utilizzando un file di configurazione globale.

Un file di configurazione può essere installato con il file SWF, creato da uno sviluppatore Flash, o aggiunto da un amministratore (per tutti gli utenti o per l'utente corrente) o da uno sviluppatore Flash (per l'utente corrente).

Per ulteriori informazioni sui file di configurazione e sul pannello Impostazioni globali di sicurezza, vedere “[Informazioni sulle impostazioni di sicurezza di Flash Player](#)” a pagina 738 e “[Specifiche dei file affidabili utilizzando Gestione impostazioni](#)” a pagina 746 e “[Creazione di file di configurazione per lo sviluppo di Flash](#)” a pagina 748.

## Prova del contenuto a livello locale con le restrizioni di sicurezza per i file locali di Flash 8

Gli sviluppatori Flash devono spesso provare le applicazioni Flash localmente. Quando l'applicazione tenta di comunicare con Internet, viene visualizzata una finestra di dialogo di richiesta. Questa finestra di dialogo potrebbe essere visualizzata quando si prova in Flash Player un file SWF che non dispone di accesso alla rete. Per ulteriori informazioni sulla pubblicazione di file SWF con livelli di autorizzazioni specificati, vedere “[Pubblicazione dei file per la distribuzione locale](#)” a pagina 743. La pubblicazione di un file SWF con una di queste opzioni significa che è possibile comunicare con la rete *o* con il file system locale.

Nel contempo, potrebbe essere necessario comunicare con il file system locale e con la rete quando viene eseguita la verifica di un documento. Poiché il nuovo modello di sicurezza potrebbe interrompere il flusso di lavoro durante la fase di creazione di applicazioni Flash, è possibile utilizzare il pannello Impostazioni globali di sicurezza in Gestione impostazioni di Flash Player per specificare quali applicazioni Flash nel computer possono sempre comunicare sia con Internet che con il file system locale. Oppure, è possibile modificare il file di configurazione per specificare le directory affidabili sul disco rigido.

Per ulteriori informazioni, consultare le sezioni seguenti:

- “[Specifica dei file affidabili utilizzando Gestione impostazioni](#)” a pagina 746
- “[Creazione di file di configurazione per lo sviluppo di Flash](#)” a pagina 748

### Specifica dei file affidabili utilizzando Gestione impostazioni

È possibile specificare quale contenuto Flash sul computer può utilizzare sempre le regole di sicurezza precedenti, aggiungendo la posizione del contenuto nel pannello Impostazioni globali di sicurezza in Gestione impostazioni di Flash Player. Dopo avere aggiunto una posizione sul computer nel pannello Sicurezza, il contenuto in quella posizione è *affidabile*. Flash Player non richiederà l'autorizzazione ed è sempre consentito l'utilizzo delle regole di sicurezza precedenti, anche se nel pannello Sicurezza è selezionata l'opzione Nega sempre. L'elenco Considera sempre affidabili i file in queste posizioni sostituisce le opzioni nel pannello Impostazioni. Vale a dire, se si sceglie di negare sempre al contenuto locale e Web il diritto di utilizzare le regole di sicurezza precedenti, ai file locali inclusi nell'elenco dei file affidabili è consentito sempre l'utilizzo delle regole precedenti.

L'elenco dei file sempre affidabili nella parte inferiore del pannello si applica specificamente al contenuto Flash scaricato nel computer, non a quello che viene utilizzato quando si visita un sito Web.

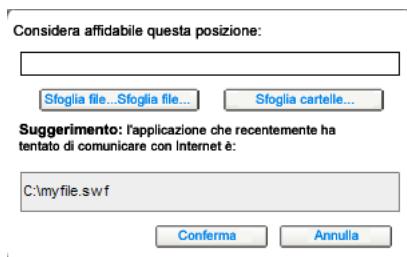
L'esempio seguente illustra come specificare che un file SWF locale può comunicare con Internet. Quando si prova un file in un browser localmente (File > Anteprima pubblicazione > HTML), potrebbe essere visualizzata una finestra di dialogo relativa alla sicurezza. Se si fa clic su Impostazioni, viene visualizzato il pannello Impostazioni globali di sicurezza di Gestione impostazioni.



#### Per specificare che un file SWF locale può comunicare con Internet e con il file system locale:

1. Nel pannello Impostazioni globali di sicurezza, fare clic sul menu a comparsa e selezionare Aggiungi.

Si apre la relativa finestra



Se si è arrivati in Gestione impostazioni facendo clic sul pulsante Impostazioni in una finestra di dialogo, la finestra Aggiungi contiene un percorso nel formato C:\nomedirectory\nameofile.swf o /Users/nomedirectory/nameofile.swf che indica il file che ha tentato di comunicare con Internet, ma è stato bloccato dalle impostazioni di sicurezza di Flash Player. Se il percorso contiene il contenuto a cui si desidera consentire la comunicazione con Internet, copiarlo e incollarlo nella casella Considera affidabile questa posizione oppure fare clic su uno dei pulsanti Sfoglia per cercare il contenuto.

È possibile aggiungere un singolo file o un'intera directory. Se si aggiunge un'intera directory, vengono considerati affidabili tutti i file e le sottodirectory che vi sono contenuti. Alcuni contenuti Flash consistono di più file correlati e potrebbe essere necessario considerare affidabile l'intera directory in cui si trovano tutti i file correlati. In generale, evitare di considerare affidabili le directory di primo livello.

**2. Fare clic su Conferma.**

Il percorso viene aggiunto al pannello Impostazioni globali di sicurezza. Ai percorsi elencati è sempre consentito l'utilizzo delle regole di sicurezza precedenti, anche se sono selezionate le opzioni Chiedi sempre o Nega sempre nella parte superiore del pannello.

Dopo aver aggiunto i percorsi affidabili, riavviare il contenuto Flash locale aggiornando la finestra del browser o riavviando il lettore.

Se si fa clic su Consenti sempre, si applica quell'impostazione solo per consentire sempre il contenuto precedente (Flash Player 7 e versioni precedenti). L'impostazione non “consente sempre” il contenuto di Flash Player 8. È preferibile specificare le applicazioni e le directory Flash presenti nel computer che possono comunicare sia con Internet che con il file system locale.

## Creazione di file di configurazione per lo sviluppo di Flash

Lo strumento di creazione di Flash 8 imposta un flag sul disco rigido che identifica lo sviluppatore e lo indirizza ad una versione specifica del pannello Impostazioni globali di sicurezza invece che ad un pannello Impostazioni globali di sicurezza orientato all'utente. Il flag si trova nel file FlashAuthor.cfg sul disco rigido che viene installato automaticamente durante l'installazione dello strumento di creazione della versione di base di Flash 8 e di Flash Professional 8.

Il file FlashAuthor.cfg si trova nelle seguenti directory approssimative:

**Windows** *disco di avvio\Documents and Settings\<nome utente>\Application Data\Macromedia\Flash Player\#Security*

**Macintosh** */Users/<nome utente>/Library/Preferences/Macromedia/Flash Player/#Security/*

Per impostazione predefinita, questo file è impostato su LocalSecurityPrompt=Author. Questo significa che le avvertenze visualizzate sul computer sono dirette ad uno sviluppatore Flash invece che ad un utente senza lo strumento di creazione installato.

È possibile provare le applicazioni locali come utente finale e visualizzare le finestre di dialogo di avvertenza che un utente finale potrebbe incontrare. A questo scopo, aprire il file FlashAuthor.cfg in un editor di testo e modificare LocalSecurityPrompt nel file FlashAuthor.cfg in modo che corrisponda alla seguente impostazione:

```
LocalSecurityPrompt=User
```

Potrebbe essere opportuno fornire un file FlashAuthor.cfg file, con LocalSecurityPrompt impostato su Author, agli altri sviluppatori nel processo di progettazione o sviluppo o agli utenti che provano le applicazioni Flash sul proprio disco rigido locale senza disporre dello strumento di creazione di Flash 8. In questo modo è possibile imitare l'esperienza dell'utente finale con il contenuto distribuito a livello locale.

**NOTA**

Il file FlashAuthor.cfg, se eliminato, viene ricreato quando si avvia lo strumento di creazione di Flash 8.

Nella directory #Security sul disco rigido, si può creare una directory FlashPlayerTrust in cui è possibile memorizzare file di configurazione univoci. All'interno di questi file, è possibile specificare directory o applicazioni da considerare affidabili sul disco rigido. Questa directory non richiede accesso amministrativo, pertanto gli utenti privi di autorizzazioni di amministratore possono impostare autorizzazioni per i file SWF e le applicazioni di prova.

Se non si specifica una directory, il contenuto potrebbe non funzionare in modo appropriato. I file di configurazione in una directory FlashPlayerTrust contengono percorsi di directory. Il file può contenere un elenco di varie directory ed è possibile aggiungere nuovi percorsi nel file. Flash Player si aspetta un solo percorso per riga nei file di configurazione. Una riga che inizia con un segno # (senza uno spazio iniziale davanti) viene considerata commento.

**Per creare un file di configurazione per considerare affidabile una directory:**

1. Individuare la cartella #Security sul disco rigido.
2. Creare una cartella denominata FlashPlayerTrust nella cartella #Security.
3. Creare un nuovo file nella directory FlashPlayerTrust utilizzando un editor di testo e salvarlo come **myTrustFiles.cfg**.  
È possibile usare un nome univoco qualsiasi per il file di configurazione.
4. Individuare la directory nella quale si provano le applicazioni Flash.
5. Digitare o incollare ogni percorso di directory (qualsiasi percorso di directory sul disco rigido) su una nuova riga nel file. È possibile incollare più percorsi di directory su righe separate. Al termine, il file è simile all'esempio seguente:  
`C:\Documents and Settings\<nome utente>\Documenti\files\  
C:\Documents and Settings\<nome utente>\Documenti\testapps\`
6. Salvare le modifiche in myTrustFiles.cfg.
7. Provare un documento che accede ai file locali e di rete dalla directory aggiunta nel file.  
Le applicazioni Flash salvate in questa directory a questo punto possono accedere ai file locali e alla rete.

Possono esservi numerosi percorsi di directory salvati in ciascun file di configurazione ed altrettanto numerosi file \*.cfg salvati nella directory FlashPlayerTrust.

Se si creano applicazioni che vengono installate sul disco rigido di un utente finale, potrebbe essere necessario creare un file di configurazione in FlashPlayerTrust per specificare una directory affidabile per l'applicazione. È possibile creare file di configurazione nella directory FlashPlayerTrust che specificano la posizione dell'applicazione affidabile. Vedere la procedura precedente per informazioni su questa directory e sulla creazione dei file di configurazione.

**NOTA**

Un programma di installazione viene eseguito da un utente con autorizzazioni di amministratore in un computer.

È necessario sviluppare uno schema di assegnazione dei nomi univoco per evitare conflitti con altre applicazioni che potrebbero installare file in questa directory. Ad esempio, per evitare conflitti è consigliabile usare il nome univoco della società e del software nel nome file.

**SUGGERIMENTO**

Se non si desidera utilizzare file di configurazione, è possibile pubblicare le applicazioni Flash su un server di prova separato anziché fornire ai client o agli altri sviluppatori i file SWF da eseguire sui loro dischi rigidi.

Per ulteriori informazioni sui file di configurazione, vedere [www.macromedia.com/go/flashauthrcfg](http://www.macromedia.com/go/flashauthrcfg). È possibile creare un unico file di configurazione per considerare affidabili una o più directory. Per informazioni dettagliate sulla sicurezza, vedere [www.macromedia.com/devnet/security/](http://www.macromedia.com/devnet/security/) e [www.macromedia.com/software/flashplayer/security/](http://www.macromedia.com/software/flashplayer/security/).

## Informazioni sulla proprietà sandboxType

La proprietà System.security.sandboxType di Flash Player 8 restituisce il tipo di funzione di sicurezza sandbox in cui opera il file SWF chiamante.

La proprietà sandboxType ha uno dei quattro valori seguenti:

**remote** Il file SWF risiede in Internet e opera secondo le regole della funzione di sicurezza sandbox domain-based.

**localTrusted** Il file SWF è un file locale che è stato definito affidabile dall'utente, utilizzando le Impostazioni globali di sicurezza di Gestione impostazioni o un file di configurazione FlashPlayerTrust. Il file SWF può sia leggere dalle origini dati locali che comunicare con la rete (ad esempio Internet).

**localWithFile** Il file SWF è un file locale che non è stato definito affidabile dall'utente e non è stato pubblicato con una designazione di connessione in rete. Il file SWF può leggere dalle origini dati locali ma non può comunicare con la rete (ad esempio Internet).

**localWithNetwork** Il file SWF è un file locale che non è stato definito affidabile dall'utente ed è stato pubblicato con l'opzione Accedi solo alla rete selezionata nella finestra di dialogo Impostazioni pubblicazione (scheda Flash). Il file SWF può comunicare con la rete ma non può leggere da origini dati locali.

È possibile verificare la proprietà `sandboxType` da qualsiasi file SWF, sebbene venga restituito un valore solo nei file pubblicati per Flash Player 8. Questo significa che quando si pubblica per Flash Player 7 o versioni precedenti, non si sa se la proprietà `sandboxType` viene supportata in fase di runtime. Se la proprietà non è supportata in fase di runtime, il valore è `undefined` e ciò si verifica quando la versione di Flash Player (indicata dalla proprietà `System.capabilities.version`) è inferiore a 8. Se il valore è `undefined`, è possibile determinare il tipo di funzione di sicurezza sandbox a seconda se l'URL del file SWF sia o meno un file locale. Se il file SWF è un file locale, Flash Player lo classifica `localTrusted` (ovvero come veniva considerato tutto il contenuto locale prima di Flash Player 8); in caso contrario Flash Player classifica il file SWF `remote`.

## Informazioni sulle restrizioni di tipo local-with-file-system

Un file local-with-file-system non è stato registrato utilizzando il file di configurazione nella directory FlashPlayerTrust, il pannello Impostazioni globali di sicurezza in Gestione impostazioni, o non gli sono state concesse autorizzazioni di rete nella finestra di dialogo Impostazioni pubblicazione nell'ambiente di creazione di Flash.



Per informazioni sulle funzioni di sicurezza sandbox, vedere “[Nozioni fondamentali sulle funzioni di sicurezza sandbox locali](#)” a pagina 737.

Questi file includono contenuto precedente che viene riprodotto in Flash Player 8. Se si sviluppa contenuto in Flash 8, o si dispone di contenuto che rientra in una delle seguenti categorie, è necessario che l'utente registri il file come affidabile. Per informazioni sulla registrazione di un file come affidabile, vedere “[Specifiche dei file affidabili utilizzando Gestione impostazioni](#)” a pagina 746. Per informazioni sulla concessione di autorizzazioni per la riproduzione di file locali mediante l'uso di file di configurazione, vedere “[Creazione di file di configurazione per lo sviluppo di Flash](#)” a pagina 748.

I file SWF di tipo local-with-file-system sono soggetti alle seguenti restrizioni:

- Non possono accedere alla rete, che include quanto segue:
  - Caricamento di altri file SWF dalla rete (*eccetto* l'utilizzo di percorsi UNC non di Internet)
  - Invio di richieste HTTP
  - Eseguire connessioni mediante XMLSocket, Flash Remoting, o NetConnection
  - Chiamare `getURL()` *eccetto* se si utilizza `getURL("file:...")` o `getURL("mailto:...")`
- Possibilità di interazione con altri file local-with-file-system ma con restrizioni alle seguenti operazioni:
  - Scambio di script (ad esempio accesso di ActionScript ad oggetti in altri file SWF).
  - Chiamare `System.security.allowDomain`
  - Utilizzo di `LocalConnection` come sender o listener e indipendentemente dai gestori `LocalConnection.allowDomain`.

**NOTA**

I file SWF di tipo local-with-file-system possono interagire con altri file SWF dello stesso tipo non di rete. Tuttavia, non possono interagire con file SWF local-with-network.

I file SWF local-with-file-system hanno accesso in lettura a file noti nel file system locale. Ad esempio, è possibile utilizzare `XML.load()` in un file SWF local-with-file-system se si carica dal file system locale e non da Internet.

- I file SWF local-with-file-system non possono comunicare con pagine HTML, che include quanto segue:
  - Creazione di script in ingresso (ad esempio ExternalInterface API, ActiveX, LiveConnect e XPConnect)
  - Creazione di script in uscita (ad esempio chiamate personalizzate a `fscommand` e `getURL("javascript:...")`)

**NOTA**

Un'eccezione alla regola si ha se la pagina HTML è affidabile.

# Informazioni su domini, sicurezza tra domini e file SWF

Per impostazione predefinita, Flash Player 7 e le versioni successive impediscono a un file SWF appartenente a un determinato dominio di leggere dati, oggetti o variabili di file SWF appartenenti a domini diversi. Il contenuto caricato tramite protocolli non sicuri (non HTTPS), inoltre, non può leggere contenuto caricato tramite un protocollo sicuro (HTTPS), anche se entrambi si trovano nello stesso dominio. Un file presente in `http://www.macromedia.com/main.swf`, ad esempio, non può caricare dati da `https://www.macromedia.com/data.txt` senza un'autorizzazione esplicita. Analogamente un file SWF appartenente a un determinato dominio non può caricare dati con l'ausilio di `loadVars()`, da un altro dominio.

Indirizzi IP numerici identici sono compatibili, tuttavia un nome di dominio non è compatibile con un indirizzo IP, anche se il nome di dominio viene risolto nello stesso indirizzo IP.

Nella tabella seguente sono riportati alcuni esempi di domini compatibili:

www.macromedia.com	www.macromedia.com
data.macromedia.com	data.macromedia.com
65.57.83.12	65.57.83.12

Nella tabella seguente sono riportati alcuni esempi di domini incompatibili:

www.macromedia.com	data.macromedia.com
macromedia.com	www.macromedia.com
www.macromedia.com	macromedia.com
65.57.83.12	www.macromedia.com (anche se questo dominio viene risolto in 65.57.83.12)
www.macromedia.com	65.57.83.12 (anche se www.macromedia.com viene risolto in questo indirizzo IP)

Flash Player 8 non consente ai file SWF locali di comunicare con Internet senza una configurazione appropriata. Per informazioni sull'impostazione di un file di configurazione per la prova di contenuto a livello locale, vedere [“Creazione di file di configurazione per lo sviluppo di Flash” a pagina 748](#).

Per ulteriori informazioni sulla sicurezza, vedere [www.macromedia.com/devnet/security/](http://www.macromedia.com/devnet/security/) e [www.macromedia.com/software/flashplayer/security/](http://www.macromedia.com/software/flashplayer/security/).

Per ulteriori informazioni, consultare i seguenti argomenti:

- “Regole del nome di dominio per le impostazioni e i dati locali” a pagina 754
- “Accesso a più domini e sottodomini tra file SWF” a pagina 754
- “Come consentire il caricamento di dati tra domini diversi” a pagina 762

## Regole del nome di dominio per le impostazioni e i dati locali

Per impostazione predefinita, in Flash Player 6 vengono utilizzate le regole di corrispondenza del superdominio per l'accesso alle impostazioni locali, ad esempio per le autorizzazioni di accesso a videocamere o microfoni oppure a dati locali quali ad esempio gli oggetti condivisi. Le impostazioni e i dati dei file SWF che risiedono nei siti here.xyz.com, there.xyz.com e xyz.com sono pertanto condivisi e sono tutti memorizzati nel sito xyz.com.

In Flash Player 7, per impostazione predefinita vengono invece utilizzate le regole di corrispondenza del dominio specifico. Quindi, le impostazioni e i dati di un file che risiede nel sito here.xyz.com vengono memorizzati in tale sito e le impostazioni e i dati di un file che risiede nel sito there.xyz.com vengono memorizzati in there.xyz.com e così via. La proprietà `System.exactSettings` consente di specificare quali regole utilizzare. Questa proprietà è supportata per i file pubblicati per Flash Player 6 e versioni successive. Per i file pubblicati per Flash Player 6, il valore predefinito è `false`, che determina l'uso delle regole di corrispondenza del superdominio. Per i file pubblicati per Flash Player 7 o 8, il valore predefinito è `true` e vengono quindi utilizzate le regole di corrispondenza del dominio specifico. Se si utilizzano le impostazioni o i dati locali persistenti e si desidera pubblicare un file SWF di Flash Player 6 per Flash Player 7 o 8, potrebbe essere necessario impostare questo valore su `false` nel file trasferito. Per ulteriori informazioni, vedere `exactSettings` (`System.exactSettings` property) nella *Guida di riferimento di ActionScript 2.0*.

## Accesso a più domini e sottodomini tra file SWF

Quando si sviluppa una serie di file SWF che comunicano tra loro in linea, ad esempio quando si utilizzano oggetti `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.loadClip()`, o Local Connection, è possibile caricare i file SWF in domini o in sottodomini diversi di un singolo superdominio.

Nei file pubblicati per Flash Player 5 o versioni precedenti, non esistevano limitazioni per l'accesso a più domini o sottodomini.

Nei file pubblicati per Flash Player 6, è possibile utilizzare il gestore `LocalConnection.allowDomain` o il metodo `System.security.allowDomain()` per specificare l'accesso autorizzato a più domini (ad esempio, per consentire l'accesso a un file del sito `someSite.com` da parte di un file del sito `someOtherSite.com`) e inoltre non sono necessari comandi per autorizzare l'accesso dei sottodomini (ad esempio è consentito l'accesso a un file del sito `www.someSite.com` da parte di un file del sito `store.someSite.com`).

I file pubblicati per Flash Player 7 implementano l'accesso tra file SWF diversamente rispetto alle versioni precedenti. La differenza è duplice: per prima cosa, Flash Player 7 implementa le regole di corrispondenza del dominio specifico anziché le regole di corrispondenza del superdominio. Pertanto, il file al quale viene eseguito l'accesso, anche se è stato pubblicato per una versione precedente a Flash Player 7, deve permettere esplicitamente l'accesso tra più domini o sottodomini. Questo argomento è trattato più avanti in questa sezione. In secondo luogo, un file che risiede in un sito che utilizza un protocollo di sicurezza (HTTPS) deve permettere esplicitamente l'accesso da parte di un file di un sito che utilizza un protocollo di tipo HTTP o FTP; questo argomento è discusso nella sezione successiva (vedere [“Accesso tra file SWF da HTTP a HTTPS” a pagina 767](#)).

Di solito si chiama `System.security.allowDomain` nelle applicazioni. Tuttavia, quando il receiver `LocalConnection` è un file SWF HTTPS e il sender non lo è, viene invece chiamato `allowInsecureDomain`.

Il problema seguente interessa solo i file SWF pubblicati per Flash Player 7. Quando il receiver è HTTPS e il sender è un file SWF locale, viene chiamato `allowDomain()`, anche se si dovrebbe chiamare `allowInsecureDomain()`. Tuttavia, in Flash Player 8, quando un receiver `LocalConnection` HTTPS è Flash Player 8 e il sender è un file locale, viene chiamato `allowInsecureDomain()`.

I file che vengono eseguiti in Flash Player 8 sono soggetti a modifiche rispetto a come venivano eseguiti in Flash Player 7. La chiamata di `System.security.allowDomain` consente solo operazioni di scambio di script in cui il file SWF a cui si accede corrisponde a quello che ha chiamato `System.security.allowDomain`. In altre parole, un file SWF che chiama `System.security.allowDomain` consente ora l'accesso solo a se stesso. Nelle versioni precedenti, la chiamata di `System.security.allowDomain` consentiva operazioni di scambio di script in cui il file SWF a cui si accede poteva essere un qualsiasi file SWF nello stesso dominio di quello che ha chiamato `System.security.allowDomain`. In tal modo veniva aperto l'intero dominio del file SWF chiamante.

È stato aggiunto il supporto per il valore carattere jolly (\*) per `System.security.allowDomain("*")` e `System.security.allowInsecureDomain("*")`. Il valore carattere jolly (\*) consente operazioni di scambio di script in cui il file che accede corrisponde a qualsiasi file e può essere caricato da qualunque posizione (ad esempio un'autorizzazione globale). Le autorizzazioni con caratteri jolly possono essere utili, ma devono essere conformi alle nuove regole di sicurezza per i file locali in Flash Player 8. In particolare, i file locali non provengono da un dominio, quindi deve essere utilizzato il valore carattere jolly. Tuttavia, il valore carattere jolly deve essere utilizzato con cautela poiché qualunque dominio ha accesso al file. Per ulteriori informazioni, vedere `allowInsecureDomain` (`security.allowInsecureDomain` method).

In alcune situazioni si carica un file SWF secondario da un dominio diverso da quello che lo chiama. Potrebbe essere utile consentirgli di attivare uno script nel file SWF principale, ma non si conosce il dominio finale da cui viene caricato il file SWF secondario. Questa situazione può verificarsi ad esempio quando si utilizzano i reindirizzamenti per il bilanciamento del carico di lavoro o i server di terze parti. In questa situazione, è possibile utilizzare la proprietà `MovieClip._url` come argomento di questo metodo. Ad esempio, se si carica un file SWF in `my_mc`, è possibile chiamare `System.security.allowDomain(my_mc._url)`. In tal caso, attendere che il file SWF in `my_mc` inizi il caricamento, poiché il valore della proprietà `_url` non è ancora definitivo e corretto. Per determinare se un file SWF secondario ha iniziato il caricamento, utilizzare `MovieClipLoader.onLoadStart`.

In realtà può verificarsi anche la situazione opposta: è possibile creare un file SWF secondario impostato per accettare gli script eseguiti dal proprio elemento principale ma di cui non conosce il dominio (ovvero è un file SWF che potrebbe essere caricato da vari domini). In tal caso, chiamare `System.security.allowDomain(_parent._url)` dal file SWF secondario. Non è necessario attendere il caricamento del file SWF principale poiché sarà già stato completato al momento del caricamento del file secondario.



Se il file SWF a cui si accede su Internet viene caricato da un URL HTTPS, quel file dovrà chiamare `System.security.allowInsecureDomain("*")`.

Nella tabella seguente sono elencate le regole di corrispondenza del dominio nelle diverse versioni di Flash Player:

File pubblicati per Flash Player	Accesso a più domini e sottodomini tra file SWF (è necessario <code>allowDomain()</code> )	Accesso a sottodomini tra file SWF
5 o precedente	Nessuna restrizione	Nessuna restrizione
6	Corrispondenza con il superdominio: <code>allowDomain()</code> è necessario se i superdomini non corrispondono.	Nessuna restrizione
7 e successive	Corrispondenza di dominio esatta Autorizzazione esplicita per l'accesso di file su siti HTTPS a file su siti HTTP o FTP	Corrispondenza di dominio esatta Autorizzazione esplicita per l'accesso di file su siti HTTPS a file su siti HTTP o FTP

**NOTA**

È necessario `System.security.allowInsecureDomain` in Flash Player 7 e versioni successive se viene eseguito l'accesso da HTTP a HTTPS, anche se si dispone di corrispondenza di dominio esatta.

Le versioni che controllano il comportamento di Flash Player sono versioni di file SWF (la versione di Flash Player specificata di un file SWF), non la versione stessa di Flash Player. Quando, ad esempio, Flash Player 8 riproduce un file SWF pubblicato per la versione 7, viene applicato un comportamento conforme alla versione 7. In questo modo, viene garantito che gli aggiornamenti del player non modificano il comportamento di `System.security.allowDomain()` dei file SWF distribuiti.

Poiché Flash Player 7 e versioni successive implementano le regole di corrispondenza del dominio specifico anziché le regole di corrispondenza del superdominio, potrebbe essere necessario modificare gli script esistenti per leggerli da file pubblicati per Flash Player 7 o 8. È comunque possibile pubblicare i file modificati per Flash Player 6. Se nei file sono state utilizzate le istruzioni `LocalConnection.allowDomain()` o `System.security.allowDomain()` e sono state specificate le regole di corrispondenza del superdominio, è necessario cambiare i parametri per specificare le regole di corrispondenza del dominio specifico. L'esempio seguente mostra le eventuali modifiche necessarie se si dispone del codice di Flash Player 6:

```
// Comandi di Flash Player 6 in un file SWF all'indirizzo Web
// www.anyOldSite.com
// per consentire l'accesso ai file SWF che risiedono all'indirizzo Web
// www.someSite.com
// oppure presso il sito Web store.someSite.com
System.security.allowDomain("someSite.com");
my_lc.allowDomain = function(sendingDomain) {
 return(sendingDomain=="someSite.com");
}
// Comandi corrispondenti per autorizzare all'accesso i file SWF
// pubblicati per Flash Player 7 o versioni successive
System.security.allowDomain("www.someSite.com", "store.someSite.com");
my_lc.allowDomain = function(sendingDomain) {
 return(sendingDomain=="www.someSite.com" ||
 sendingDomain=="store.someSite.com");
}
```

Potrebbe inoltre essere necessario aggiungere ai file istruzioni analoghe alle seguenti se già non sono in uso. Se, ad esempio, il file SWF risiede nel sito `www.someSite.com` e si desidera consentire l'accesso a un file SWF pubblicato per Flash Player 7 all'indirizzo `store.someSite.com`, è necessario aggiungere istruzioni come quelle seguenti al file presente all'indirizzo `www.someSite.com` (è comunque possibile pubblicare il file all'indirizzo `www.someSite.com` per Flash Player 6):

```
System.security.allowDomain("store.someSite.com");
my_lc.allowDomain = function(sendingDomain) {
 return(sendingDomain=="store.someSite.com");
}
```

Tenere inoltre presente che, se un'applicazione Flash Player 6 eseguita all'interno di Flash Player 7 tenta di accedere a dati all'esterno del relativo dominio, vengono applicate le regole di corrispondenza del dominio di Flash Player 7 e versioni successive e all'utente viene chiesto di concedere o negare l'accesso.

Per riepilogare, potrebbe essere necessario modificare i file aggiungendo o cambiando le istruzioni `allowDomain` se si pubblicano file per Flash Player 7 o versioni successive che soddisfano le seguenti condizioni:

- È stato implementato lo scambio di script tra file SWF (vedere “[Come consentire l'accesso ai dati tra file SWF di domini diversi](#)” a pagina 760).
- Il file SWF chiamato (di qualsiasi versione) non risiede in un sito che utilizza un protocollo di sicurezza (HTTPS) oppure i file SWF chiamanti e chiamati risiedono entrambi in siti HTTPS. Se solo il file SWF chiamato utilizza il protocollo HTTPS, vedere “[Accesso tra file SWF da HTTP a HTTPS](#)” a pagina 767.)
- I file SWF non si trovano nello stesso dominio, ad esempio uno si trova all'indirizzo `www.domain.com` e l'altro all'indirizzo `store.domain.com`.

È necessario apportare le seguenti modifiche:

- Se il file SWF chiamato è pubblicato per Flash Player 7 o versioni successive, includere `System.security.allowDomain` o `LocalConnection.allowDomain` nel file SWF chiamato utilizzando la corrispondenza esatta dominio-nome .
- Se il file SWF chiamato è pubblicato per Flash Player 6, modificare il file chiamato aggiungendo o cambiando un'istruzione `System.security.allowDomain` o `LocalConnection.allowDomain`, utilizzando la corrispondenza dominio-nome, come mostrato negli esempi di codice precedenti di questa sezione. È possibile pubblicare il file modificato per Flash Player 6 o per Flash Player 7.
- Se il file SWF chiamato è pubblicato per Flash Player 5 o versioni precedenti, trasferire il file chiamato in Flash Player 6 o 7 e aggiungere un'istruzione `System.security.allowDomain` utilizzando la corrispondenza esatta dominio-nome, come mostrato negli esempi di codice precedenti in questa sezione. Gli oggetti `LocalConnection` non sono supportati in Flash Player 5 o versioni precedenti.

Per informazioni sulle funzioni di sicurezza sandbox locali, vedere “[Informazioni sulla sicurezza dei file locali e Flash Player](#)” a pagina 735.

## Come consentire l'accesso ai dati tra file SWF di domini diversi

Due file SWF devono appartenere allo stesso dominio per poter accedere ai relativi dati (variabili e oggetti). Per impostazione predefinita, affinché i file possano condividere dati, in Flash Player 7 e nelle versioni successive i due domini devono corrispondere esattamente. Un file SWF, tuttavia, può concedere l'accesso a file SWF appartenenti a domini specifici chiamando `LocalConnection.allowDomain` o `System.security.allowDomain()`.

`System.security.allowDomain()` consente ai file SWF e HTML in domini specificati di accedere agli oggetti e alle variabili nel file SWF che contiene la chiamata `allowDomain()`.

Se due file SWF appartengono allo stesso dominio, ad esempio `http://mysite.com/movieA.swf` e `http://mysite.com/movieB.swf`, mediante `movieA.swf` è possibile esaminare e modificare variabili, oggetti, proprietà, metodi e così via in `movieB.swf`, e mediante `movieB` è possibile procedere allo stesso modo per `movieA`. Questa operazione è detta scambio di script tra filmati, o *scambio di script*.

Se due file SWF appartengono a domini diversi, ad esempio `http://mysite.com/movieA.swf` e `http://othersite.com/movieB.swf`, per impostazione predefinita non è consentito a `movieA.swf` di eseguire script su `movieB.swf` né viceversa. Se si chiama

`System.security.allowDomain("mysite.com")`, `movieB.swf` concede a `movieA.swf` l'autorizzazione a eseguire script su `movieB.swf`. Un file SWF concede a file SWF di altri domini l'autorizzazione a eseguire script su tale file SWF mediante la chiamata a `System.security.allowDomain()`. Questa operazione è detta *script tra domini*.

Per ulteriori informazioni su `System.security.allowDomain()`, scambio di script e script tra domini, vedere `allowDomain` (`security.allowDomain` method) nella *Guida di riferimento di ActionScript 2.0*.

Si supponga, ad esempio, che `main.swf` appartenga al dominio `www.macromedia.com` e che carichi un altro file SWF (`data.swf`) da `data.macromedia.com` in un'istanza di un clip filmato creato in modo dinamico con l'ausilio di `createEmptyMovieClip()`.

```
// In macromedia.swf
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadMovie("http://data.macromedia.com/data.swf");
```

Si supponga che nella linea temporale principale di `data.swf` sia stato definito il metodo `getData()`. Per impostazione predefinita, `main.swf` non può chiamare il metodo `getData()` definito in `data.swf` dopo il caricamento del file, perché i due file SWF non appartengono allo stesso dominio. La chiamata del metodo seguente in `main.swf` dopo il caricamento di `data.swf`, ad esempio, non riesce:

```
// In macromedia.swf dopo il caricamento di data.swf:
target_mc.getData(); // La chiamata di questo metodo avrà esito negativo
```

Il file data.swf, però, può consentire l'accesso ai file SWF appartenenti a www.macromedia.com con l'ausilio del gestore LocalConnection.allowDomain e del metodo System.security.allowDomain(), a seconda del tipo di accesso richiesto. Il codice seguente aggiunto a data.swf consente a un file appartenente a www.macromedia.com di accedere alle variabili e ai metodi presenti nel codice stesso:

```
// In data.swf
this._lockroot = true;
System.security.allowDomain("www.macromedia.com");
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String):Boolean {
 return (sendingDomain == "www.macromedia.com");
};
function getData():Void {
 var timestamp:Date = new Date();
 output_txt.text += "data.swf:" + timestamp.toString() + "\n\n";
}
output_txt.text = "**INIT**:\n\n";
```

A questo punto la funzione getData nel file SWF caricato potrà essere chiamata dal file macromedia.swf. Tenere presente che allowDomain consente a tutti i file SWF del dominio specificato di eseguire script su tutti gli altri file SWF del dominio per permettere l'accesso solo se il file SWF a cui si accede non appartiene a un sito che utilizza un protocollo sicuro (HTTPS).

Per ulteriori informazioni sulla corrispondenza dei nomi di dominio, vedere [“Accesso a più domini e sottodomini tra file SWF”](#) a pagina 754.

## File di criteri server-side per l'accesso ai dati

Un documento Flash può caricare dati da un'origine esterna utilizzando una delle seguenti chiamate di caricamento dati: XML.load(), XML.sendAndLoad(), LoadVars.load(), LoadVars.sendAndLoad(), loadVariables(), loadVariablesNum(), MovieClip.loadVariables(), XMLSocket.connect() e Macromedia Flash Remoting (NetServices.createGatewayConnection). Inoltre, in fase di runtime, un file SWF può importare librerie condivise in runtime o elementi definiti in un altro file SWF. Per impostazione predefinita, i dati o RSL devono trovarsi nello stesso dominio del file SWF che sta caricando i dati o i media esterni.

Per rendere disponibili dati ed elementi di librerie condivise in runtime per file SWF residenti in domini differenti, utilizzare un *file di criteri per i domini*. Si tratta di un file XML che fornisce al server la possibilità di indicare che i propri dati e documenti sono disponibili per i file SWF provenienti da specifici domini o da tutti i domini. A qualsiasi file SWF servito da un dominio specificato dal file di criteri del server è consentito l'accesso a dati, risorse o RSL provenienti da tale server.

Se si caricano dati esterni, è necessario creare file di criteri anche se non si desidera trasferire alcun file in Flash Player 7. Se si utilizzano RSL, è necessario creare file di criteri se il file chiamante o chiamato è pubblicato per Flash Player 7.

Per ulteriori informazioni, consultare i seguenti argomenti:

- “Come consentire il caricamento di dati tra domini diversi” a pagina 762
- “Informazioni sui percorsi personalizzati dei file di criteri” a pagina 764
- “Informazioni sui file di criteri XMLSocket” a pagina 765

## Come consentire il caricamento di dati tra domini diversi

Quando un documento Flash tenta di accedere a dati da un dominio diverso, Flash Player tenta automaticamente di caricare un file di criteri dal quel dominio. Se il dominio del documento Flash che tenta di accedere ai dati è compreso nel file dei criteri, i dati sono accessibili automaticamente.

I file di criteri devono essere denominati crossdomain.xml e possono essere presenti nella directory principale o in un'altra directory del server che fornisce i dati con codice ActionScript aggiuntivo. Vedere “[Informazioni sui percorsi personalizzati dei file di criteri](#) a pagina 764”. Questi file funzionano solo sui server che utilizzano i protocolli HTTP, HTTPS o FTP per la comunicazione. Il file di criteri è specifico per la porta e il protocollo del server su cui è presente.

Un file di criteri che si trova, ad esempio, all'indirizzo <https://www.macromedia.com:8080/> crossdomain.xml sarà valido solo per le chiamate per il caricamento di dati eseguite a [www.macromedia.com](http://www.macromedia.com) tramite HTTPS sulla porta 8080.

Un'eccezione a questa regola è rappresentata dall'uso dell'oggetto XMLSocket per la connessione con un server socket in un dominio diverso. In questo caso, un server HTTP eseguito sulla porta 80 dello stesso dominio del server socket deve fornire il file di criteri per la chiamata al metodo.

Un file di criteri XML contiene un solo tag <cross-domain-policy> che, a sua volta, contiene zero o più tag <allow-access-from>. Ogni tag <allow-access-from> contiene un attributo, domain che specifica un indirizzo IP esatto, un dominio esatto o un dominio carattere jolly (ovvero qualsiasi dominio). I domini carattere jolly sono indicati da un solo asterisco (\*), che corrisponde a tutti i domini e a tutti gli indirizzi IP oppure da un asterisco seguito da un suffisso che corrisponde a tutti i domini che terminano con il suffisso specificato. I suffissi devono iniziare con un punto. I domini carattere jolly con suffissi possono tuttavia essere rappresentati da domini costituiti solo dal suffisso senza il punto iniziale, ad esempio, foo.com è considerato parte di \*.foo.com. I caratteri jolly non sono consentiti quando vengono specificati domini IP.

Se si specifica un indirizzo IP, l'accesso è consentito solo ai file SWF caricati da quell'indirizzo IP utilizzando la sintassi IP (ad esempio http://65.57.83.12/flashmovie.swf) e non ai file caricati utilizzando la sintassi del nome del dominio. Flash Player non esegue la risoluzione DNS.

Nell'esempio seguente è illustrato un file di criteri che consente l'accesso a documenti Flash appartenenti a foo.com, www.friendOffFoo.com, \*.foo.com e 105.216.0.40 da un documento Flash appartenente a foo.com:

```
<?xml version="1.0"?>
<!- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
 <allow-access-from domain="www.friendOfFoo.com" />
 <allow-access-from domain="*.foo.com" />
 <allow-access-from domain="105.216.0.40" />
</cross-domain-policy>
```

È inoltre possibile consentire l'accesso a documenti appartenenti a qualsiasi dominio, come illustrato di seguito:

```
<?xml version="1.0"?>
<!- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
 <allow-access-from domain="*" />
</cross-domain-policy>
```

Ogni tag <allow-access-from> dispone dell'attributo opzionale secure che per impostazione predefinita ha valore true. È possibile impostarlo su false se il file di criteri si trova su un server HTTPS e si desidera consentire ai file SWF di un server HTTP di caricare dati dal server HTTPS.

L'impostazione dell'attributo secure su false potrebbe compromettere la sicurezza garantita dal protocollo HTTPS.

Se il file SWF che si scarica proviene da un server HTTPS, ma il file SWF che lo carica si trova su un server HTTP, sarà necessario aggiungere l'attributo `secure="false"` al tag `<allow-access-from>`, come illustrato nel codice seguente:

```
<allow-access-from domain="www.foo.com" secure="false" />
```

Utilizzare un file di criteri che non contiene tag `<allow-access-from>` equivale a non utilizzare alcun criterio su un server.

## Informazioni sui percorsi personalizzati dei file di criteri

Flash Player 7 (7.0.19.0) supporta un metodo denominato `System.security.loadPolicyFile` che consente di specificare un percorso personalizzato su un server per un file di criteri dei domini che non deve quindi necessariamente trovarsi nella directory principale. Con Flash Player 7 (7.0.14.0) la ricerca dei file di criteri viene eseguita solo nella directory principale di un server. L'amministratore del sito potrebbe trovare pratico tuttavia inserire il file in una directory diversa. Per ulteriori informazioni sul metodo `loadPolicyFile` e sulle connessioni `XMLSocket`, vedere “[Informazioni sui file di criteri XMLSocket](#)” a pagina 765 e `loadPolicyFile` (`security.loadPolicyFile` method) nella *Guida di riferimento di ActionScript 2.0*.

Se viene utilizzato il metodo `loadPolicyFile`, è possibile inserire il file di criteri in qualunque directory se i file SWF che devono utilizzare il file chiamano `loadPolicyFile` per indicare a Flash Player il percorso del file di criteri. I file di criteri che non vengono inseriti nella directory principale hanno però un'area di validità limitata, ovvero consentono l'accesso solo ai percorsi corrispondenti o situati a un livello inferiore rispetto a quello della gerarchia del server in cui si trovano.

Il metodo `loadPolicyFile` è disponibile solo in Flash Player 7 (7.0.19.0) o versioni successive. Gli autori dei file SWF che utilizzano il metodo `loadPolicyFile` devono effettuare una delle seguenti operazioni:

- Utilizzare Flash Player 7 (7.0.19.0) o le versioni successive e sviluppare per queste versioni.
- Verificare che il sito da cui i dati provengono disponga di un file di criteri nel percorso predefinito (la directory principale) o in un percorso diverso. Con le versioni precedenti di Flash Player viene utilizzato il percorso predefinito.

In caso contrario, il caricamento di dati tra domini diversi non riesce.

**ATTENZIONE**

Se il file SWF utilizza `loadPolicyFile`, i visitatori che dispongono di Flash Player 6 o versione precedente o di Flash Player 7 (7.0.19.0) o versione successiva non riscontreranno problemi. I visitatori che dispongono di Flash Player 7 (7.0.14.0) non possono avvalersi del supporto di `loadPolicyFile`.

Se si desidera utilizzare un file di criteri in un percorso personalizzato sul server, chiamare `System.security.loadPolicyFile` *prima* di eseguire richieste che dipendono dal file di criteri, come illustrato di seguito:

```
System.security.loadPolicyFile("http://www.foo.com/folder1/folder2/
 crossdomain.xml");
var my_xml:XML = new XML();
my_xml.load("http://www.foo.com/folder1/folder2/myData.xml");
```

È possibile caricare diversi file di criteri con aree di validità sovrapposte con l'ausilio di `loadPolicyFile`. Per tutte le richieste, Flash Player tenta di consultare tutti i file la cui area di validità comprende il percorso della richiesta. Se non è possibile garantire l'accesso ai dati da domini diversi con uno dei file di criteri, può essere possibile garantirlo con un altro file. Se tutti i tentativi di accesso falliscono, Flash Player cerca nel percorso predefinito del file `crossdomain.xml`, vale a dire nella directory principale. La richiesta fallisce definitivamente se nel percorso predefinito non sono presenti file di criteri.

## Informazioni sui file di criteri XMLSocket

A ogni tentativo di connessione XMLSocket, Flash Player (7.0.14.0) cerca `crossdomain.xml` sulla porta 80 nel sottodominio del server HTTP in cui viene effettuato il tentativo di connessione. Flash Player 7 (7.0.14.0) e tutte le versioni precedenti limitano le connessioni XMLSocket alle porte 1024 e superiori. In Flash Player 7 (7.0.19.0) e nelle versioni successive, invece, ActionScript è in grado di indicare a Flash Player un percorso non predefinito per il file di criteri, utilizzando `System.security.loadPolicyFile`. Tutti i percorsi personalizzati dei file di criteri XMLSocket devono trovarsi sempre su un server socket XML.

Nell'esempio seguente, Flash Player recupera un file di criteri da un URL specificato:

```
System.security.loadPolicyFile("http://www.foo.com/folder/policy.xml");
```

Qualsiasi autorizzazione concessa dal file di criteri per quel percorso viene applicata a tutto il contenuto che si trova allo stesso livello o a un livello inferiore nella gerarchia del server. Se quindi si tenta di caricare i seguenti dati, è possibile caricarne solo alcuni da determinati percorsi:

```
myLoadVars.load("http://foo.com/sub/dir/vars.txt"); // allowed
myLoadVars.load("http://foo.com/sub/dir/deep/vars2.txt"); // allowed
myLoadVars.load("http://foo.com/elsewhere/vars3.txt"); // not allowed
```

Come soluzione provvisoria, è possibile caricare più file di criteri in un solo file SWF con l'ausilio di `loadPolicyFile`. Flash Player attende sempre che venga completato lo scaricamento dei file di criteri prima di negare l'autorizzazione a una richiesta che coinvolge file di criteri. Se nel file SWF non vengono autorizzati altri criteri, Flash Player cerca nel percorso predefinito di `crossdomain.xml`.

Una speciale sintassi consente il recupero dei file di criteri direttamente da un server XMLSocket:

```
System.security.loadPolicyFile("xmlsocket://foo.com:414");
```

In questo esempio, Flash Player cerca di recuperare un file di criteri dall'host e dalla porta specificati. È possibile utilizzare qualsiasi porta se il file di criteri non si trova nella directory predefinita, ovvero la directory principale. In caso contrario, è possibile utilizzare solo la porta 1024 e le porte superiori (come avveniva con le versioni precedenti dei lettori). Dopo aver stabilito una connessione con la porta specificata, Flash Player invia `<policy-file-request />` seguito da un byte nullo.

Il server socket XML potrebbe essere configurato per gestire i file di criteri nei modi seguenti:

- Gestione dei file di criteri e delle normali connessioni socket sulla stessa porta. Il server deve attendere il comando `<policy-file-request />` prima di trasmettere un file di criteri.
- Gestione dei file di criteri su una porta diversa da quella utilizzata per le normali connessioni. In questo caso il server può inviare un file di criteri non appena viene stabilita una connessione con la porta dedicata ai file di criteri.

Il server deve inviare un byte nullo per terminare un file di criteri prima di chiudere la connessione. Se la connessione non viene chiusa, Flash Player la interrompe alla ricezione del byte nullo di terminazione.

Un file di criteri gestito da un server socket XML ha la stessa sintassi di qualsiasi altro file di criteri, con la differenza che questo file specifica anche le porte a cui viene garantito l'accesso. Le porte consentite vengono specificate in un attributo `to-ports` del tag `<allow-access-from>`. Se un file di criteri è associato a una porta inferiore a 1024, può autorizzare l'accesso a qualsiasi porta. Se un file di criteri proviene dalla porta 1024 o da una porta superiore, può autorizzare l'accesso solo alle porte superiori alla 1024. È possibile specificare numeri di porte specifici, intervalli di porte e caratteri jolly. Di seguito è riportato un esempio di un file di criteri XMLSocket:

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

Poiché la possibilità di eseguire la connessione alle porte inferiori a 1024 è disponibile in Flash Player 7 (7.0.19.0) e versioni successive, l'uso di un file di criteri caricato mediante `loadPolicyFile` è sempre obbligatorio per autorizzare questa operazione, anche se un file SWF si connette al proprio sottodomainio.

## Accesso tra file SWF da HTTP a HTTPS

È necessario utilizzare un gestore o un metodo `allowDomain` per consentire l'accesso a un file SWF di un dominio da parte di un file SWF di un altro dominio. Se però il file SWF a cui si accede appartiene a un sito che utilizza un protocollo sicuro (HTTPS), il gestore o il metodo `allowDomain` non consentono l'accesso da parte di un file SWF presente in un sito che utilizza un protocollo non sicuro. A questo scopo occorre utilizzare le istruzioni `LocalConnection.allowInsecure Domain()` o `System.security.allowInsecureDomain()`. Per ulteriori informazioni, vedere “[Come consentire l'accesso tra file da HTTP a HTTPS](#)” a pagina 768.

## Come consentire l'accesso tra file da HTTP a HTTPS

Oltre all'implementazione delle regole di corrispondenza del dominio specifico, è ora necessario autorizzare esplicitamente l'accesso ai file dei siti che utilizzano un protocollo sicuro (HTTPS) da parte dei file che invece utilizzano un protocollo non sicuro. A seconda che il file chiamato sia pubblicato per Flash Player 6, 7 o 8, è necessario implementare una delle istruzioni `allowDomain` (vedere [“Accesso a più domini e sottodomini tra file SWF a pagina 754”](#), oppure utilizzare le nuove istruzioni `LocalConnection.allowInsecureDomain` o `System.security.allowInsecureDomain()`.

Per consentire, ad esempio, l'accesso al file SWF `https://www.someSite.com/data.swf` da parte di un file SWF presente all'indirizzo `http://www.someSite.com`, aggiungere il codice seguente a `data.swf`:

```
// In data.swf
System.security.allowInsecureDomain("www.someSite.com");
my_lc.allowInsecureDomain = function(sendingDomain:String):Boolean {
 return (sendingDomain == "www.someSite.com");
};
```



L'implementazione di un'istruzione `allowInsecureDomain()` compromette la sicurezza garantita dal protocollo HTTPS. Queste modifiche devono essere eseguite solo se non è possibile riorganizzare il sito in modo che i file utilizzino tutti il protocollo HTTPS.

Nel codice riportato di seguito è contenuto un esempio delle eventuali modifiche necessarie:

```
// Comandi di un file SWF di Flash Player 6 all'indirizzo https://
// www.someSite.com
// per consentire l'accesso tramite i file SWF di Flash Player 7 che
// risiedono
// all'indirizzo http://www.someSite.com o http://www.someOtherSite.com
System.security.allowDomain("someOtherSite.com");
my_lc.allowDomain = function(sendingDomain) {
 return(sendingDomain=="someOtherSite.com");
}
// Comandi corrispondenti in un file SWF di Flash Player 7
// per consentire l'accesso tramite i file SWF di Flash Player 7 che
// risiedono
// all'indirizzo http://www.someSite.com o http://www.someOtherSite.com
System.security.allowInsecureDomain("www.someSite.com",
"www.someOtherSite.com");
my_lc.allowInsecureDomain = function(sendingDomain) {
 return(sendingDomain=="www.someSite.com" ||
 sendingDomain=="www.someOtherSite.com");
}
```

Potrebbe inoltre essere necessario aggiungere ai file istruzioni analoghe alle seguenti se già non sono in uso. Una modifica potrebbe essere richiesta anche se entrambi i file si trovano nello stesso dominio, ad esempio se un file all'indirizzo `http://www.domain.com` chiama un file all'indirizzo `https://www.domain.com`.

Per riepilogare, potrebbe essere necessario modificare i file aggiungendo o cambiando le istruzioni se i file pubblicati per Flash Player 7 o versione successiva soddisfano le seguenti condizioni:

- È stato implementato lo script SWF, utilizzando gli oggetti `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` o Local Connection.
- Il file che esegue la chiamata non risiede su un dominio con protocollo HTTPS e il file chiamato utilizza il protocollo HTTPS.

È necessario apportare le seguenti modifiche:

- Se il file è pubblicato per Flash Player 7, includere le istruzioni `System.security.allowInsecureDomain` o `LocalConnection.allowInsecureDomain` nel file chiamato utilizzando la corrispondenza esatta dominio-nome come mostrato negli esempi di codice precedenti in questa sezione.
- Se il file chiamato è pubblicato per Flash Player 6 o una versione precedente e sia il file chiamante che quello chiamato si trovano nello stesso dominio, ad esempio se un file all'indirizzo `http://www.domain.com` chiama un file all'indirizzo `https://www.domain.com`, non è necessario apportare alcuna modifica.
- Se il file chiamato è pubblicato per Flash Player 6, i file non si trovano nello stesso dominio e non si desidera trasferire il file chiamato in Flash Player 7, aggiungere o modificare nel file chiamato un'istruzione `System.security.allowDomain` o `LocalConnection.allowDomain` utilizzando la corrispondenza esatta dominio-nome, come mostrato nei precedenti esempi di codice in questa sezione.
- Se il file chiamato è pubblicato per Flash Player 6 e si desidera trasferire il file chiamato in Flash Player 7, includervi l'istruzione `System.security.allowInsecureDomain` o `LocalConnection.allowInsecureDomain` utilizzando la corrispondenza esatta dominio-nome, come mostrato nei precedenti esempi di codice in questa sezione.

- Se il file chiamato è pubblicato per Flash Player 5 o una versione precedente e i due file non si trovano nello stesso dominio, sono disponibili due alternative. È possibile trasferire il file chiamato in Flash Player 6 e aggiungere o modificare un'istruzione `System.security.allowDomain` utilizzando la corrispondenza esatta dominio-nome, come mostrato nei precedenti esempi di codice in questa sezione, oppure è possibile trasferire il file chiamato in Flash Player 7 e includere un'istruzione `System.security.allowInsecureDomain` nel file chiamato utilizzando la corrispondenza esatta dominio-nome, come mostrato nei precedenti esempi di codice in questa sezione.

# Esecuzione del debug delle applicazioni

18

Macromedia Flash Basic 8 e Macromedia Flash Professional 8 forniscono numerosi strumenti per la prova del codice ActionScript contenuto nei file SWF. Il Debugger consente di trovare errori in un file SWF durante la sua esecuzione in Flash Debug Player (vedere “[Esecuzione del debug degli script](#)” a pagina 771). Flash inoltre fornisce i seguenti ulteriori strumenti di debug:

- Il Pannello Output, che visualizza i messaggi di errore (tra cui alcuni errori di runtime) e gli elenchi di variabili e oggetti (vedere “[Uso del pannello Output](#)” a pagina 785).
- L'istruzione `trace`, che invia note di programmazione e valori di espressioni al pannello Output (vedere “[Uso dell'istruzione trace](#)” a pagina 789).
- Le istruzioni `throw` e `try..catch..finally` consentono di provare e rispondere agli errori di runtime dallo script.

In questa sezione, viene descritto come eseguire il debug degli script e delle applicazioni Flash mediante il Debugger e come utilizzare il pannello Output. Per ulteriori informazioni, consultare i seguenti argomenti:

<a href="#">Esecuzione del debug degli script</a> .....	771
<a href="#">Uso del pannello Output</a> .....	785

## Esecuzione del debug degli script

Il Debugger in Flash 8 consente di trovare errori nel file SWF durante la sua esecuzione in Flash Player. Il file SWF deve essere visualizzato in una versione speciale di Flash Player denominata *Flash Debug Player*. Quando si installa lo strumento di creazione del codice, Flash Debug Player viene installato automaticamente. Pertanto, se si installa Flash e si visita un sito Web che presenta contenuto Flash, oppure si esegue l'opzione Prova filmato, di fatto si sta utilizzando Flash Debug Player. È anche possibile eseguire il programma di installazione nella directory seguente in Windows o Macintosh: *Directory di installazione di Flash\Players\Debug\* cartella o avviare il Flash Debug Player autonomo dalla stessa directory.

Quando si utilizza il comando Controllo > Prova filmato per provare file SWF che implementano i controlli da tastiera (spostamento tramite tasto Tab, tasti di scelta rapida creati con `Key.addListener()` e così via), selezionare Controllo > Disattiva scelte rapide da tastiera. Questa opzione garantisce che la pressione dei tasti di scelta rapida nell'ambiente di creazione esegua esattamente la funzione loro assegnata. Ad esempio, nell'ambiente di creazione Ctrl+U apre la finestra di dialogo Preferenze. Se con lo script la combinazione Ctrl+U viene assegnata a un'azione che sottolinea il testo sullo schermo, quando si utilizza Prova filmato, la pressione di Ctrl+U consente di aprire la finestra di dialogo Preferenze, anziché di eseguire l'azione di sottolineatura. Per fare in modo che il comando Ctrl+U venga passato al lettore, selezionare Controllo > Disattiva scelte rapide da tastiera.

**ATTENZIONE**

Se viene utilizzata un'applicazione non in lingua italiana in un sistema in lingua italiana, il comando Prova filmato non viene completato correttamente se in una parte del file SWF sono presenti caratteri che non è possibile rappresentare con lo schema di codifica MBCS. Ad esempio, i percorsi in lingua giapponese su sistemi in lingua italiana non sono validi. Tutte le aree dell'applicazione che utilizzano il lettore esterno sono soggette a questa limitazione.

Il Debugger visualizza un elenco gerarchico dei clip filmato attualmente caricati in Flash Player. Questa utilità consente di visualizzare e modificare i valori delle proprietà e delle variabili durante la riproduzione del file SWF e di utilizzare punti di interruzione per fermare il file e scorrere singolarmente le righe di codice ActionScript.

È possibile utilizzare il Debugger in modalità di prova con i file locali o residenti su un server Web remoto. Il Debugger consente di impostare punti di interruzione in ActionScript per fermare Flash Player e scorrere le righe del codice in esecuzione. È quindi possibile tornare agli script e modificarli per ottenere i risultati desiderati.

Una volta attivato il Debugger, la relativa barra di stato visualizza l'URL o il percorso locale del file, indica se il file viene eseguito in modalità di prova o da una postazione remota, oltre a visualizzare in tempo reale l'elenco di visualizzazione del clip filmato. Quando si aggiungono o rimuovono clip filmato dal file, l'elenco di visualizzazione viene immediatamente aggiornato con le modifiche. Per modificare le dimensioni dell'elenco, è sufficiente spostare la barra di divisione orizzontale.

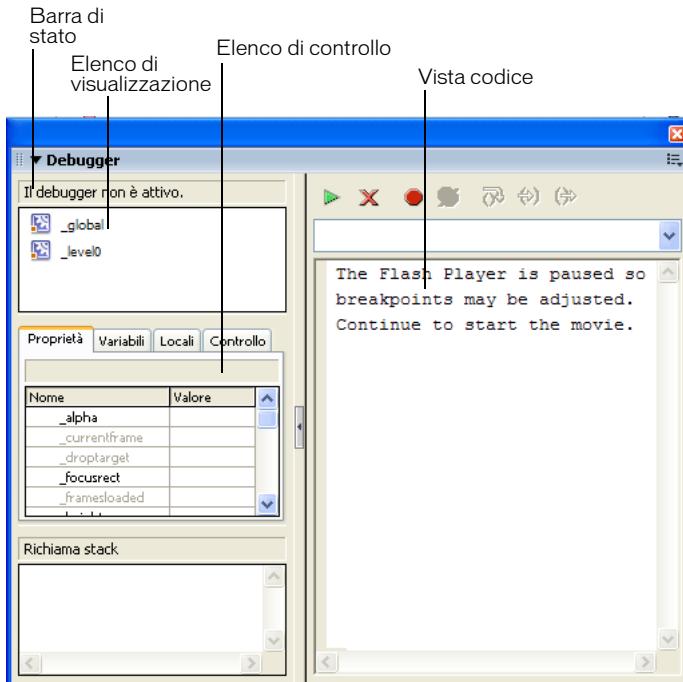
**Per attivare il Debugger in modalità di prova filmato:**

- Selezionare Controllo > Debug filmato.

Questo comando consente di esportare il file SWF con le informazioni di debug (il file SWD) e di attivare il debug per il file SWF, aprendo il Debugger e il file SWF in modalità di prova.

**NOTA**

Se necessario, è possibile ridimensionare le varie aree del pannello Debugger. Quando il puntatore cambia aspetto tra un'area e l'altra, è possibile trascinare per ridimensionare l'elenco di visualizzazione, l'elenco di controllo e la vista codice.



Per ulteriori informazioni, consultare gli argomenti seguenti:

- “Debug di un file SWF da una postazione remota”
- “Visualizzazione e modifica delle variabili” a pagina 776
- “Uso dell’elenco di controllo” a pagina 777
- “Visualizzazione delle proprietà dei clip filmato e modifica delle proprietà modificabili” a pagina 779
- “Impostazione e rimozione dei punti di interruzione” a pagina 780
- “Informazioni sulle operazioni con le righe di codice” a pagina 783

## Debug di un file SWF da una postazione remota

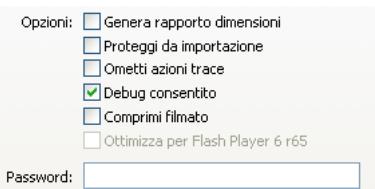
È possibile eseguire il debug di un file SWF remoto utilizzando la versione autonoma, ActiveX o Plug-In di Flash Player. Per individuare queste versioni di Flash Player, sfogliare la directory seguente in Windows o Macintosh: *Directory di installazione di Flash\Players\Debug*.

Quando si esporta un file SWF, è possibile attivare l'operazione di debug nel file e creare una password di debug apposita. Il Debugger non si attiva automaticamente.

Per consentire soltanto ad alcuni utenti di visualizzare i file SWF con Flash Debug Player, è possibile pubblicare il file impostando una password di debug. Come JavaScript o HTML, anche ActionScript consente di visualizzare le variabili client-side. Per memorizzare variabili in modo sicuro inviarle a un'applicazione su un server, anziché memorizzarle nel file. È tuttavia possibile che lo sviluppatore del filmato Flash non desideri rivelare altre informazioni riservate, quali la struttura del clip filmato. Una password di debug permette di proteggere il proprio lavoro.

### Per attivare il debug remoto di un file SWF:

1. Scegliere File > Impostazioni pubblicazione.
2. Nella scheda Flash della finestra di dialogo Impostazioni pubblicazione, selezionare Debug consentito.



3. Eventualmente, immettere una password nell'apposita casella.  
In seguito, sarà possibile scaricare le informazioni sul Debugger solo inserendo la password. Se non si desidera attivare la protezione tramite password, è sufficiente lasciare il campo vuoto.
4. Chiudere la finestra di dialogo Impostazioni pubblicazione e selezionare uno dei comandi seguenti:
  - Selezionare Controllo > Debug filmato.
  - Selezionare File > Esporta > Esporta filmato.
  - Selezionare File > PubblicaFlash crea un file di debug con estensione .swd e lo salva nella stessa directory del file SWF. Il file SWD contiene informazioni sull'uso dei punti di interruzione e consente di scorrere il codice per eseguire il debug di ActionScript.

5. È necessario inserire il file SWD nella stessa directory del file SWF sul server.

In caso contrario, è comunque possibile eseguire il debug in modalità remota, tuttavia il Debugger non dispone di informazioni sui punti di interruzione e non è quindi possibile scorrere il codice.

6. In Flash, selezionare Finestra > Debugger.

7. Nel Debugger, selezionare Attiva debug remoto dal menu a comparsa (nell'angolo superiore destro del pannello).

**Per attivare il Debugger da una postazione remota:**

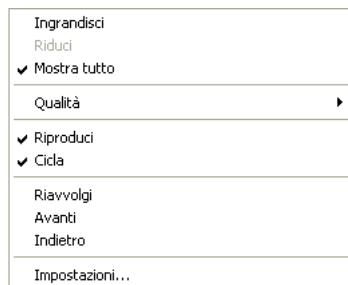
1. Aprire l'applicazione di creazione Flash.
2. In un browser o nella versione di debug del lettore autonomo, aprire il file SWF pubblicato dalla postazione remota.

Viene visualizzata la finestra di dialogo Debug remoto.



**NOTA**

Se la finestra Debug remoto non viene visualizzata, il file SWD non è stato individuato. In questo caso, fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) nel file SWF per visualizzare il menu di scelta rapida, quindi selezionare Debugger.



- 3.** Nella finestra di dialogo Debug remoto, selezionare il computer host locale o un altro dispositivo:
  - Selezionare il computer host locale se Debug Player e l'applicazione di creazione Flash si trovano sullo stesso computer.
  - Selezionare un altro dispositivo se Debug Player e l'applicazione di creazione Flash non si trovano sullo stesso computer. Immettere l'indirizzo IP del computer su cui viene eseguita l'applicazione di creazione Flash.

- 4.** Stabilità la connessione, viene richiesta la password.

Immettere la password di debug se ne è stata impostata una.

Nel Debugger appare l'elenco di visualizzazione del file SWF. Se il file SWF non viene riprodotto, è possibile che l'esecuzione del Debugger sia stata sospesa. Fare clic su Continua per riavviarla.

## Visualizzazione e modifica delle variabili

Nella scheda Variabili del Debugger sono visualizzati i nomi e i valori delle variabili globali e della linea temporale presenti nel file SWF selezionato nell'elenco di visualizzazione. Se si modifica il valore di una variabile nella scheda Variabili, è possibile osservare l'effetto della modifica direttamente nel file SWF in esecuzione. Ad esempio, per provare il rilevamento della presenza di collisioni in un gioco è possibile immettere il valore di una variabile in modo da posizionare una palla nel punto appropriato accanto a una parete.

Nella scheda Locali del Debugger sono visualizzati i nomi e i valori delle variabili locali disponibili nella riga di codice ActionScript in corrispondenza della quale è stata interrotta l'esecuzione del file SWF, vale a dire un punto di interruzione o un altro punto all'interno di una funzione definita dall'utente.

### Per visualizzare una variabile:

1. Selezionare il clip filmato contenente la variabile dall'elenco di visualizzazione.

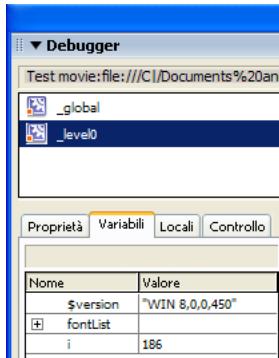
Per visualizzare le variabili globali, selezionare il clip `_global` nell'elenco di visualizzazione.



Se necessario, è possibile ridimensionare le varie aree del pannello Debugger. Quando il puntatore cambia aspetto tra un'area e l'altra, è possibile trascinare per ridimensionare l'elenco di visualizzazione, l'elenco di controllo e la vista codice.

- 2.** Fare clic sulla scheda Variabili.

L'elenco di visualizzazione viene aggiornato automaticamente durante la riproduzione del file SWF. Se si rimuove un clip filmato dal file SWF in corrispondenza di un fotogramma specifico, dall'elenco di visualizzazione del Debugger vengono rimossi anche la variabile e il nome di variabile. Se tuttavia una variabile viene contrassegnata per l'elenco di controllo (vedere “[Uso dell'elenco di controllo](#)” a pagina 777), la variabile viene rimossa dalla scheda Variabili, ma può essere visualizzata nella scheda Controllo.



#### Per modificare il valore di una variabile:

- Fare doppio clic sul valore, quindi immetterne uno nuovo.

Il valore non deve essere un'espressione. Ad esempio, è possibile usare "Ciao", 3523 o "http://www.macromedia.com" ma non `x + 2` o `eval("nome:" + i)`. Il valore può essere una stringa, ovvero qualsiasi valore racchiuso tra virgolette [""], un numero o un valore booleano (`true` or `false`).



Per visualizzare il valore di un'espressione nel pannello Output in modalità di prova, utilizzare l'istruzione `trace`. Vedere “[Uso dell'istruzione trace](#)” a pagina 789.

## Uso dell'elenco di controllo

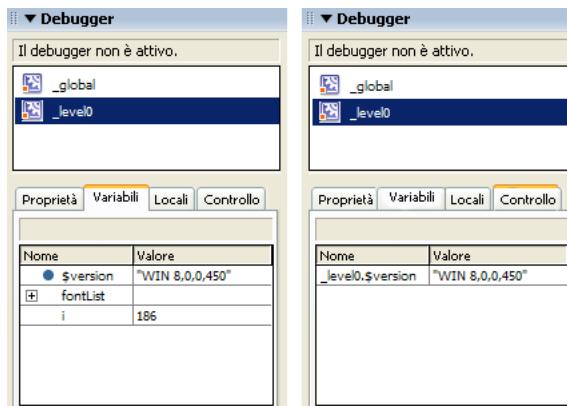
Per monitorare in modo organizzato un insieme di variabili critiche, è possibile contrassegnare le variabili affinché vengano visualizzate nell'elenco di controllo. L'elenco di controllo visualizza il percorso assoluto della variabile e il suo valore. Inoltre, come nella scheda Variabili, anche nell'elenco di controllo è possibile immettere un nuovo valore per la variabile. Nell'elenco di controllo possono essere visualizzate solo variabili e proprietà a cui è possibile accedere tramite un percorso target assoluto, ad esempio `_global` o `_root`.

Se all'elenco di controllo si aggiunge una variabile locale, il valore viene visualizzato solo quando Flash Player viene fermato in corrispondenza della riga di ActionScript relativa alla variabile. Tutte le altre variabili vengono visualizzate durante la riproduzione del file SWF. Se il Debugger non è in grado di individuare il valore della variabile, questo viene classificato come non definito.

**NOTA**

Se necessario, è possibile ridimensionare le varie aree del pannello Debugger. Quando il puntatore cambia aspetto tra un'area e l'altra, è possibile trascinare per ridimensionare l'elenco di visualizzazione, l'elenco di controllo e la vista codice.

Nell'elenco di controllo non possono essere visualizzate né proprietà né funzioni, ma solo variabili.



Variabili contrassegnate per l'inserimento nell'elenco di controllo e variabili presenti nell'elenco

**Per aggiungere variabili all'elenco di controllo, effettuare una delle seguenti operazioni:**

- Nella scheda Variabili o Locali, fare clic con il pulsante destro del mouse (Windows) oppure fare clic tenendo premuto il tasto Ctrl (Macintosh) su una variabile selezionata dal menu di scelta rapida. La variabile viene contrassegnata da un punto blu.
- Nella scheda Controllo, fare clic con il pulsante destro del mouse (Windows) oppure fare clic tenendo premuto il tasto Ctrl (Macintosh), quindi selezionare Aggiungi dal menu di scelta rapida. Fare doppio clic sulla colonna del nome e specificare il percorso target del nome della variabile nel campo.

**Per rimuovere variabili dall'elenco di controllo:**

- Nella scheda Controllo nella scheda Variabili, fare clic con il pulsante destro del mouse (Windows) oppure fare clic tenendo premuto il tasto Ctrl (Macintosh) e scegliere Rimuovi dal menu di scelta rapida.

## Visualizzazione delle proprietà dei clip filmato e modifica delle proprietà modificabili

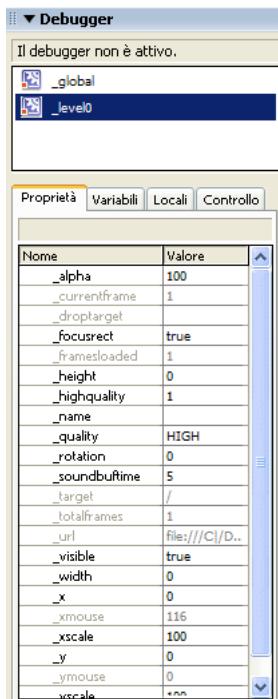
Nella scheda Proprietà del Debugger sono visualizzati tutti i valori delle proprietà dei clip filmato presenti sullo stage. Durante l'esecuzione del file SWF, è possibile modificare un valore e vedere gli effetti della modifica. Alcune proprietà dei clip filmato sono di sola lettura e non possono essere modificate.

**NOTA**

Se necessario, è possibile ridimensionare le varie aree del pannello Debugger. Quando il puntatore cambia aspetto tra un'area e l'altra, è possibile trascinare per ridimensionare l'elenco di visualizzazione, l'elenco di controllo e la vista codice.

### Per visualizzare le proprietà di un clip filmato nel Debugger:

1. Selezionare un clip filmato dall'elenco di visualizzazione.
2. Fare clic sulla scheda Proprietà nel Debugger.



### Per modificare il valore di una proprietà:

- Fare doppio clic sul valore, quindi immetterne uno nuovo.

Il valore non deve essere un'espressione. Ad esempio, è possibile immettere 50 o "clearwater", ma non `x + 50`. Il valore può essere una stringa, ovvero qualsiasi valore racchiuso tra virgolette [""], un numero o un valore booleano (true or false). Non è possibile immettere valori di un oggetto o di una matrice (ad esempio, `{id: "sedia"}` o `[1, 2, 3]`) nel Debugger.

**NOTA**

Per visualizzare il valore di un'espressione nel pannello Output in modalità di prova, utilizzare l'istruzione `trace`. Vedere “[Uso dell'istruzione trace](#)” a pagina 789.

## Impostazione e rimozione dei punti di interruzione

Un punto di interruzione consente di interrompere un'applicazione Flash in esecuzione in Flash Debug Player in corrispondenza di una riga specifica del codice ActionScript. È possibile utilizzare i punti di interruzione per individuare i possibili elementi critici nel codice. Ad esempio, se dopo aver specificato un gruppo di istruzioni `if..else if` non è possibile determinare quale di queste è in esecuzione, è possibile aggiungere un punto di interruzione davanti alle istruzioni e scorrerle singolarmente con il Debugger.

È possibile impostare i punti di interruzione nel pannello Azioni, nella finestra Script o nel Debugger. I punti di interruzione impostati nel pannello Azioni vengono salvati con il file FLA, mentre quelli impostati nel Debugger e nella finestra Script sono validi solo per la sessione di debug in corso.

**ATTENZIONE**

Se dopo avere impostato punti di interruzione nel pannello Azioni o nella finestra Script si sceglie il pulsante Formattazione automatica, alcuni punti di interruzione potrebbero non trovarsi più nella posizione corretta. In seguito alla formattazione del codice, parti di codice ActionScript potrebbero essere spostate su una riga diversa, perché a volte vengono rimosse le righe vuote. Potrebbe pertanto essere necessario verificare e modificare i punti di interruzione dopo la formattazione automatica. In alternativa, formattare gli script prima di impostare i punti di interruzione.

**Per impostare o rimuovere un punto di interruzione nel pannello Azioni o nella finestra Script durante una sessione di debug, effettuare una delle seguenti operazioni:**

- Fare clic sul margine sinistro. Un punto rosso indica un punto di interruzione.
- Fare clic sul pulsante Opzioni debug al di sopra del riquadro dello script.



- Fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) per visualizzare il menu di scelta rapida e selezionare Imposta punto di interruzione, Elimina punto di interruzione o Elimina i punti di interruzione in questo file.

**NOTA**

Nella finestra Script, è anche possibile selezionare Elimina i punti di interruzione in tutti i file AS.

- Premere Ctrl+Maiusc+B (Windows) oppure Comando+Maiusc+B (Macintosh).

**NOTA**

In alcune versioni precedenti di Flash, facendo clic sul margine sinistro del riquadro dello script si selezionava la riga di codice; ora si aggiunge o si elimina un punto di interruzione. Per selezionare una riga di codice, utilizzare Ctrl-clic (Windows) o Comando-clic (Macintosh).

**Per impostare ed eliminare i punti di interruzione nel Debugger, effettuare una delle seguenti operazioni:**

- Fare clic sul margine sinistro. Un punto rosso indica un punto di interruzione.
- Fare clic sul pulsante Attiva/disattiva punto di interruzione o Elimina tutti i punti di interruzione, sopra la vista codice.
- Fare clic con il pulsante destro del mouse (Windows) o fare clic tenendo premuto il tasto Ctrl (Macintosh) per visualizzare il menu di scelta rapida e selezionare Imposta punto di interruzione, Elimina punto di interruzione o Elimina tutti i punti di interruzione in questo file.
- Premere Ctrl+Maiusc+B (Windows) oppure Comando+Maiusc+B (Macintosh).

Quando Flash Player si ferma in corrispondenza di un punto di interruzione, è possibile verificare l'esecuzione della funzione, eseguire una verifica interna o eseguire la verifica dell'uscita dalla funzione presente nella riga di codice. Vedere “[Informazioni sulle operazioni con le righe di codice](#)” a pagina 783.

È possibile impostare i punti di interruzione nella finestra Script e mostrarli nel Debugger se esso ha lo stesso percorso del file ActionScript come quello aperto nella finestra Script. In modo analogo, è possibile impostare i punti di interruzione nel Debugger durante una sessione di debug e visualizzarli nel file ActionScript se aperto nella finestra Script.

**NOTA**

Non impostare punti di interruzione in corrispondenza di commenti o righe vuote, perché verrebbero ignorati.

## Informazioni sul file XML dei punti di interruzione

Quando si eseguono operazioni con punti di interruzione presenti in un file di script nella finestra Script, il file AsBreakpoints.xml consente di memorizzare le informazioni relative ai punti di interruzione. Il file AsBreakpoints.xml viene scritto nella directory Impostazioni locali, nelle posizioni seguenti:

Windows:

*disco rigido\Documents and Settings\utente\Impostazioni locali\Dat applicazioni\Macromedia\Flash 8\lingua\Configuration\Debugger\*

Macintosh:

*Macintosh HD/Utenti/nomeutente/Library/Supporto Applicazioni/Macromedia Flash 8/ Configuration/Debugger/*

Il seguente è un esempio di file AsBreakpoints.xml:

```
<?xml version="1.0"?>
<flash_breakpoints version="1.0">
 <file name="c:\tmp\myscript.as">
 <breakpoint line="10"></breakpoint>
 <breakpoint line="8"></breakpoint>
 <breakpoint line="6"></breakpoint>
 </file>
 <file name="c:\tmp\myotherscript.as">
 <breakpoint line="11"></breakpoint>
 <breakpoint line="7"></breakpoint>
 <breakpoint line="4"></breakpoint>
 </file>
</flash_breakpoints>
```

Il file XML è costituito dai tag seguenti:

**flash\_breakpoints** Questo nodo ha un attributo denominato `version` che indica la versione del file XML. La versione di Flash Player 8 è 1.0.

**file** Un nodo secondario di `flash_breakpoints`. Questo nodo ha un attributo denominato `name` che indica il nome del file contenente i punti di interruzione.

**breakpoint** Un nodo secondario di `file`. Questo nodo ha un attributo denominato `line` che indica il numero della riga in cui si trova il punto di interruzione.

Il file AsBreakpoints.xml è rosso quando si avvia Flash e viene rigenerato quando l'applicazione viene chiusa. Il file AsBreakpoints.xml viene utilizzato per tenere traccia dei punti di interruzione tra le sessioni di sviluppo di Flash. I punti di interruzione vengono mantenuti da una struttura di dati interna mentre si impostano ed eliminano durante lo sviluppo in Flash.

## Informazioni sulle operazioni con le righe di codice

Quando si avvia una sessione di debug, l'esecuzione di Flash Player viene sospesa per consentire di attivare e disattivare i punti di interruzione. Se si impostano punti di interruzione nel pannello Azioni, fare clic su Continua per riprodurre il file SWF fino al raggiungimento di un punto di interruzione. Se non sono stati impostati punti di interruzione nel pannello Azioni, è possibile utilizzare il menu di collegamento nel Debugger per selezionare uno degli script del file SWF. Dopo aver selezionato lo script, è possibile aggiungervi i punti di interruzione.

Dopo aver aggiunto i punti di interruzione, fare clic su Continua per avviare il file SWF. Quando raggiunge il punto di interruzione, il Debugger si ferma. Ad esempio, si supponga che nel codice seguente sia stato impostato un punto di interruzione all'interno di un pulsante sulla riga `myFunction():`

```
on(press){
 myFunction();
}
```

Premendo il pulsante viene raggiunto il punto di interruzione e Flash Player viene messo in pausa. A questo punto è possibile portare il Debugger sulla prima riga di `myFunction()`, in qualunque posizione sia definita nel documento. È inoltre possibile scorrere o uscire dalla funzione.

Durante lo scorrimento delle righe del codice, i valori delle variabili e delle proprietà cambiano sia nell'elenco di controllo, sia nella schede Variabili, Locali e Proprietà. Una freccia gialla sul lato sinistro del codice nel Debugger indica la riga in corrispondenza della quale è stato arrestato il Debugger. Utilizzare i pulsanti seguenti situati nella parte superiore della vista codice:



**Verifica esecuzione** fa avanzare il Debugger (indicato dalla freccia gialla) in una funzione. Verifica esecuzione è compatibile solo con le funzioni definite dall'utente.

Nell'esempio seguente, se si inserisce un punto di interruzione alla riga 7 e si fa clic su Verifica esecuzione, il Debugger avanza alla riga 2; un ulteriore clic su Verifica esecuzione fa passare alla riga 3. Se si fa clic su Verifica esecuzione in corrispondenza di righe prive di funzioni definite dall'utente, il Debugger avanza di una riga di codice. Ad esempio, se il Debugger si trova sulla riga 2 e si seleziona Verifica esecuzione, il Debugger avanza fino alla riga 3, come nell'esempio seguente:

```
1 function myFunction() {
2 x = 0;
3 y = 0;
4 }
5
6 mover = 1;
7 myFunction();
8 mover = 0;
```



I numeri presenti in questo codice rappresentano numeri di riga e non fanno parte del codice.

**Verifica uscita** consente di far avanzare il Debugger fino alla fine di una funzione. Il pulsante funziona solo se attualmente ci si trova in una funzione definita dall'utente; la freccia gialla avanza fino alla riga successiva a quella in cui era stata richiamata la funzione. Nell'esempio precedente, se si inserisce un punto di interruzione in corrispondenza della riga 3 e si fa clic su Verifica uscita, il Debugger si sposta alla riga 8. La selezione di Verifica uscita in una riga non compresa all'interno di una funzione definita dall'utente equivale alla selezione di Continua. Ad esempio, se ci si interrompe alla riga 6 e si fa clic su Verifica uscita, il lettore continua l'esecuzione dello script fino a un punto di interruzione.

**Verifica internamente** fa avanzare il Debugger oltre una riga di codice. Questo pulsante consente di spostare la freccia gialla alla riga successiva dello script. Nell'esempio precedente, se l'interruzione avviene alla riga 7 e si fa clic su Verifica internamente, si passa direttamente alla riga 8 senza scorrere `myFunction()`, sebbene il codice di `myFunction()` venga eseguito.

**Continua** consente di spostarsi dalla riga in corrispondenza della quale è stata interrotta l'esecuzione e di raggiungere il punto di interruzione successivo.

**Interrompi debug** disattiva il Debugger ma continua a riprodurre il file SWF in Flash Player.

# Uso del pannello Output

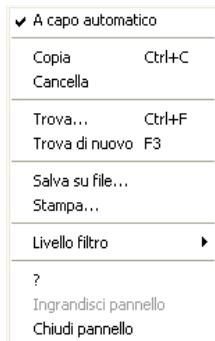
In modalità di prova, nel pannello Output sono visualizzate le informazioni che consentono di risolvere i problemi relativi al file SWF. Alcune informazioni, ad esempio gli errori di sintassi, vengono visualizzate automaticamente. Altre informazioni sono visualizzabili tramite i comandi Elenca oggetti ed Elenca variabili. (Vedere “[Elenco degli oggetti di un file SWF](#)” a pagina 787 e “[Elenco delle variabili di un file SWF](#)” a pagina 787).

Se si utilizza l'istruzione `trace` negli script, è possibile inviare informazioni specifiche al pannello Output durante l'esecuzione del file SWF. Tali informazioni possono includere note sullo stato del file SWF o sul valore di un'espressione. Vedere “[Uso dell'istruzione trace](#)” a pagina 789.

**Per visualizzare o nascondere il pannello Output, effettuare una delle operazioni seguenti:**

- Selezionare Finestra > Output.
- Premere F2.

Per visualizzare le opzioni e utilizzare il contenuto del pannello Output, fare clic sul menu a comparsa nell'angolo superiore destro.



Nella tabella seguente sono elencate le opzioni disponibili nel menu a comparsa del pannello Output:

Voce di menu	Funzione
A capo automatico	Determina se le righe troppo lunghe vengono mandate a capo automaticamente affinché l'utente non debba utilizzare la barra di scorrimento orizzontale per visualizzare tutta la riga. Se l'opzione è selezionata, le righe vanno a capo automaticamente. In caso contrario, non vanno a capo.
Copia	Copia negli Appunti tutto il contenuto del pannello Output. Per copiare solo una selezione, selezionare l'area che si desidera copiare, quindi scegliere Copia.
Cancella	Cancella tutto l'output del pannello Output.
Trova	Consente di aprire una finestra di dialogo per cercare una parola chiave o una frase all'interno del contenuto del pannello Output.
Trova di nuovo	Consente di cercare l'istanza successiva di una parola chiave o di una frase nel contenuto del pannello Output.
Salva su file	Consente di salvare il contenuto del pannello Output in un file di testo esterno.
Stampa	Consente di visualizzare la finestra di dialogo Stampa per stampare il contenuto del pannello Output su una stampante installata o su programmi installati quali Flash Paper o Acrobat.
Livello filtro	Consente di selezionare due possibili livelli di output, Nessuno o Dettagliato. Se si seleziona Nessuno, non viene inviato alcun output al browser.
Ingrandisci pannello	Consente di ingrandire il pannello Output quando è agganciato.
Chiudi pannello	Consente di chiudere il pannello Output, cancellandone il contenuto.

Per ulteriori informazioni sul pannello Output, consultare i seguenti argomenti:

- “Elenco degli oggetti di un file SWF” a pagina 787
- “Elenco delle variabili di un file SWF” a pagina 787
- “Informazioni sulla visualizzazione delle proprietà del campo di testo per il debug” a pagina 788
- “Uso dell'istruzione trace” a pagina 789
- “Aggiornamento di Flash Player per la verifica” a pagina 790

## Elenco degli oggetti di un file SWF

Nella modalità di prova, il comando Elenca oggetti consente di visualizzare gerarchicamente il livello, il fotogramma, il tipo di oggetto (forma, clip filmato o pulsante), i pulsanti target, i nomi di istanza dei clip filmato, i pulsanti e i campi di testo. Questa opzione è particolarmente utile per ottenere il percorso target e il nome dell'istanza corretti. A differenza di quanto avviene nel Debugger, l'elenco non viene aggiornato automaticamente durante la riproduzione del file SWF. Ogni volta che si desidera inviare informazioni al pannello Output è necessario selezionare il comando Elenca oggetti.

### ATTENZIONE

La selezione del comando Elenca oggetti cancella tutte le informazioni visualizzate nel pannello Output. Per non perdere queste informazioni, prima di selezionare questo comando scegliere Salva su file dal menu a comparsa Opzioni del pannello Output oppure copiare e incollare le informazioni in un altro punto.

Il comando Elenca oggetti non elenca tutti gli oggetti di dati ActionScript. Nel presente contesto per oggetto si intende una forma o un simbolo nello stage.

#### Per visualizzare un elenco degli oggetti presenti in un file SWF:

1. Se il file SWF non è eseguito in modalità di prova, selezionare Controllo > Prova filmato.
2. Selezionare Debug > Elenca oggetti.

Nel pannello Output viene visualizzato un elenco di tutti gli oggetti presenti sullo stage, come nell'esempio seguente:

```
Level #0: Frame=1 Label="Scene_1"
 Button: Target="_level0.myButton"
 Shape:
 Movie Clip: Frame=1 Target="_level0.myMovieClip"
 Shape:
 Edit Text: Target="_level0.myTextField" Text="This is sample text."
```

## Elenco delle variabili di un file SWF

In modalità di prova, il comando Elenca variabili consente di visualizzare un elenco di tutte le variabili presenti nel file SWF. Questo elenco è particolarmente utile per ottenere il percorso target e il nome della variabile. A differenza del Debugger, l'elenco non si aggiorna automaticamente durante la riproduzione del file SWF: ogni volta che si desidera inviare informazioni al pannello Output è necessario selezionare il comando Elenca variabili.

Questo comando consente anche di visualizzare le variabili globali dichiarate tramite l'identificatore `_global`. Le variabili globali vengono visualizzate all'inizio dell'elenco delle variabili, nella sezione corrispondente e ciascuna variabile è preceduta dal prefisso `_global`.

Inoltre, il comando Elenca variabili consente di visualizzare le proprietà di tipo getter/setter create tramite il metodo `Object.addProperty()` e di richiamare i metodi `get` o `set`. Una proprietà di tipo getter/setter viene visualizzata insieme a tutte le altre proprietà dell'oggetto a cui appartiene. Per distinguere facilmente queste proprietà da altre variabili, il valore di una proprietà getter/setter viene fatto precedere dalla stringa `[getter/setter]`. Il valore visualizzato per una proprietà getter/setter è determinato dalla valutazione della funzione `get` della proprietà.

**ATTENZIONE**

La selezione del comando Elenca variabili cancella tutte le informazioni visualizzate nel pannello Output. Per non perdere queste informazioni, prima di selezionare questo comando scegliere Salva su file dal menu a comparsa Opzioni del pannello Output oppure copiare e incollare le informazioni in un altro punto.

#### Per visualizzare un elenco delle variabili in un file SWF:

1. Se il file SWF non è eseguito in modalità di prova, selezionare Controllo > Prova filmato.
2. Selezionare Debug > Elenco variabili.

Nel pannello Output vengono elencate tutte le variabili contenute nel file SWF, come illustrato nell'esempio seguente:

```
Global Variables:
 Variable _global.mycolor = "lime_green"
Level #0:
 Variable _level0.$version = "WIN 7,0,19,0"
 Variable _level0.myArray = [object #1, class 'Array'] [
 0:"socks",
 1:"gophers",
 2:"mr.claw"
]
 Movie Clip: Target="_level0.my_mc"
```

#### Informazioni sulla visualizzazione delle proprietà del campo di testo per il debug

Per acquisire informazioni sul debug in relazione a oggetti `TextField`, è possibile utilizzare il comando Debug > Elenca variabili nella modalità di prova del filmato. Per la visualizzazione degli oggetti `TextField` nel pannello Output vengono adottate le seguenti convenzioni:

- Se una proprietà non viene trovata nell'oggetto, non viene visualizzata.
- Non è possibile visualizzare più di quattro proprietà per riga.
- Una proprietà con valore stringa viene visualizzata su una riga separata.

- Se dopo l'elaborazione delle proprietà incorporate vengono rilevate ulteriori proprietà definite per l'oggetto, queste vengono aggiunte alla visualizzazione utilizzando le regole riportate in precedenza al secondo e terzo punto.
- Le proprietà relative al colore vengono visualizzate come numeri esadecimali (0x00FF00).
- Le proprietà sono visualizzate nel seguente ordine: variable, text, htmlText, html, textWidth, textHeight, maxChars, borderColor, backgroundColor, textColor, border, background, wordWrap, password, multiline, selectable, scroll, hscroll, maxscroll, maxhscroll, bottomScroll, type, embedFonts, restrict, length, tabIndex, autoSize.

Durante la modalità di prova, il comando Elenca oggetti del menu Debug consente di elencare gli oggetti TextField. Se viene specificato un nome di istanza per un campo di testo, nel pannello Output viene visualizzato il percorso target completo, che include il nome dell'istanza, nel formato seguente:

```
Target = "target path"
```

Per ulteriori informazioni sul comando Elenca variabili o Elenca oggetti, consultare "[Uso del pannello Output](#)" a pagina 785.

## Uso dell'istruzione trace

L'istruzione `trace` utilizzata in uno script consente di inviare informazioni al pannello Output. Durante la prova di un file SWF o di una scena, ad esempio, è possibile inviare al pannello note di programmazione specifiche oppure impostare la visualizzazione di risultati specifici in seguito alla pressione di un pulsante o alla riproduzione di un fotogramma. L'istruzione `trace` è simile all'istruzione `alert` di JavaScript.

Quando si utilizza l'istruzione `trace` in uno script, è possibile utilizzare le espressioni come parametri. Il valore di un'espressione viene visualizzato nel pannello Output in modalità di prova, come illustrato nel frammento di codice seguente e nell'immagine del pannello Output.

**Per utilizzare l'istruzione trace in uno script:**

1. Selezionare il fotogramma 1 della linea temporale e, nel pannello Azioni, aggiungere il codice seguente:

```
this.createEmptyMovieClip("img_mc", 10);
var mcListener:Object = new Object();
mcListener.onLoadInit = function(target_mc:MovieClip) {
 trace(target_mc+" loaded in "+getTimer()+" ms");
};
mcListener.onLoadError = function(target_mc:MovieClip,
 errorCode:String, httpStatus:Number) {
 trace(">> error downloading image into "+target_mc);
 trace(">>\t errorCode="+errorCode+", httpStatus="+httpStatus);
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/404.jpg",
 img_mc);
```

2. Selezionare Controllo > Prova filmato per provare il file SWF.

Il pannello Output visualizza il messaggio seguente:



## Aggiornamento di Flash Player per la verifica

È possibile scaricare l'ultima versione di Flash Player dal sito Web del Centro di assistenza Macromedia all'indirizzo [www.macromedia.com/support/flash](http://www.macromedia.com/support/flash) e utilizzarla per provare i file SWF.

# Procedure ottimali e convenzioni di codifica per ActionScript 2.0

19

È importante che i designer e gli sviluppatori Macromedia Flash scrivano il codice e creino la struttura delle applicazioni in modo tale da risultare intuitivi e utili anche per eventuali altri partecipanti allo stesso progetto. Questo aspetto è particolarmente importante per i file FLA che contengono numerosi elementi o file di codice estesi. Quando si seguono le procedure ottimali e le convenzioni di codifica, tutti gli appartenenti alle funzioni di progettazione e di sviluppo sono in grado di comprendere la struttura dei file e il codice ActionScript e possono lavorare in modo più efficiente. Questo documento permette di formalizzare lo sviluppo e il processo di scrittura del codice in Flash.

Se, come spesso accade, allo stesso progetto Flash collaborano più designer o sviluppatori, i vantaggi che derivano dall'applicazione di un insieme predefinito di regole generali per l'uso di Flash, l'organizzazione dei file FLA e la scrittura di codice ActionScript 2.0 sono innegabili e considerevoli. Nelle sezioni di questo capitolo vengono presentate le procedure ottimali per la scrittura di codice ActionScript e alcune sezioni di *Uso di Flash* spiegano le procedure ottimali per l'uso dello strumento di creazione Flash.

Le linee guida seguenti garantiscono l'uniformità per i nuovi utenti che apprendono l'uso di Flash e la scrittura di codice ActionScript. Le procedure ottimali illustrate di seguito dovranno essere applicate in modo costante e uniforme da designer e sviluppatori, sia per il lavoro in team che per il lavoro autonomo.

- Se si lavora su documenti Flash o ActionScript

L'adozione di procedure coerenti ed efficienti permette di accelerare il flusso di lavoro. Lo sviluppo in base a convenzioni di codifica definite è più rapido perché, se occorre apportare modifiche in seguito, risulta più semplice comprendere e ricordare la struttura del documento. Il codice risultante è spesso più portabile all'interno della struttura di un progetto di grandi dimensioni e più semplice da riutilizzare.

- Se si condividono file FLA o AS

Tutti gli utenti che modificano il documento saranno in grado di trovare e comprendere rapidamente il codice ActionScript, modificarlo in modo coerente e trovare e modificare le risorse.

- Se si lavora su applicazioni

Più autori possono lavorare su un'applicazione come meno conflitti e maggiore efficienza.

Se si seguono le procedure ottimali e le convenzioni di codifica, gli amministratori del progetto o del sito possono gestire e strutturare applicazioni o progetti complessi con un minor numero di conflitti o sovrapposizioni.

- Se si apprende o si insegna l'uso di Flash e ActionScript

Se si creano applicazioni attenendosi alle procedure ottimali e seguendo le convenzioni di codifica, è meno probabile che si debbano apprendere di nuovo determinate metodologie.

Se gli studenti imparano a utilizzare Flash strutturando il codice in modo coerente, imparano a gestire il linguaggio con maggiore rapidità e facilità.

La coerenza delle tecniche utilizzate e le linee guida presentate di seguito rappresentano un valido aiuto per chi impara a utilizzare Flash o per chi deve lavorare con efficienza in team. L'uso di metodi coerenti consente di ricordare più facilmente in che modo è stato strutturato il documento, anche in caso di lavoro autonomo, specialmente quando non si utilizza il file FLA da diverso tempo.

Esistono molti altri motivi per apprendere e adottare procedure ottimali e molti altri risulteranno evidenti leggendo le indicazioni presentate di seguito e mettendole in pratica in modo regolare. Gli argomenti seguenti servono come linee guida per il lavoro in Flash che possono essere adottate integralmente o parzialmente, oppure adattate alle proprie modalità di lavoro. Molte delle linee guida presentate in questo capitolo facilitano l'adozione di una modalità di lavoro coerente per l'uso di Flash e la scrittura di codice ActionScript.

In questo capitolo vengono illustrati i seguenti argomenti sulle convenzioni di codifica e sulle procedure ottimali:

<a href="#">Convenzioni di denominazione</a> .....	793
<a href="#">Uso di commenti nel codice</a> .....	804
<a href="#">Convenzioni di codifica ActionScript</a> .....	807
<a href="#">Ottimizzazione di ActionScript e di Flash Player</a> .....	824
<a href="#">Formattazione della sintassi ActionScript</a> .....	825

# Convenzioni di denominazione

In genere, l'80% del tempo di sviluppo viene dedicato al debug, alla risoluzione dei problemi e a operazioni di manutenzione generica, specialmente per i progetti di grandi dimensioni. Anche quando si lavora su progetti più piccoli, una parte significativa del tempo viene dedicata solitamente all'analisi e alla correzione del codice. La leggibilità del codice è importante sia per lo sviluppatore che scrive il codice, sia per gli altri membri del team.

Quando si seguono determinate convenzioni di denominazione, viene migliorata la leggibilità e, di conseguenza, aumenta il flusso di lavoro e la capacità di rilevamento e correzione degli errori nel codice. Tutti i programmati seguono infatti una procedura standardizzata per la scrittura del codice, e ciò consente di migliorare il progetto sotto diversi aspetti.

L'uso delle convenzioni di assegnazione dei nomi delle variabili può fornire i seguenti vantaggi:

- Migliorano la leggibilità del codice, consentendo di identificare immediatamente il tipo di dati di una variabile. Questo aspetto può risultare molto utile sia per gli studenti che apprendono la scrittura del codice, sia per gli sviluppatori che non hanno familiarità con il codice.
- Sono facili da cercare e sostituire, se necessario.
- Contribuiscono a ridurre i conflitti con le parole riservate e i costrutti del linguaggio.
- Possono essere di aiuto per distinguere tra variabili da aree di validità diverse (variabili locali, proprietà di classi, parametri e così via).

Le sezioni seguenti contengono le linee guida consigliate per l'assegnazione dei nomi per la scrittura di codice ActionScript, come l'assegnazione di nomi a file, variabili, costanti, componenti e così via. “[Formattazione della sintassi ActionScript](#)” a pagina 825 descrive le convenzioni di formattazione specifiche di ActionScript e comuni ad altri linguaggi di programmazione. “[Convenzioni di codifica ActionScript](#)” a pagina 807 descrive le convenzioni di codifica specifiche per la scrittura di script di ActionScript e per lo sviluppo con Flash 8.

**NOTA**

Flash Player 7 e 8 non sono del tutto conformi alla specifica del linguaggio ECMAScript (ECMA-262) Edizione 3. Per consultare la specifica e le informazioni sul funzionamento del linguaggio, (vedere [www.ecma-international.org/publications/standards/Ecma-262.htm](http://www.ecma-international.org/publications/standards/Ecma-262.htm)).

In questa sezione vengono descritti i seguenti argomenti:

- “Linee guida generali per l’assegnazione di nomi” a pagina 794
- “Evitare parole riservate o costrutti del linguaggio” a pagina 795
- “Assegnazione di nomi alle variabili” a pagina 796
- “Assegnazione di nomi alle costanti” a pagina 799
- “Assegnazione di nomi alle variabili booleane” a pagina 799
- “Assegnazione di nomi a funzioni e metodi” a pagina 800
- “Assegnazione di nomi a classi e oggetti” a pagina 800
- “Assegnazione di nomi ai pacchetti” a pagina 802
- “Assegnazione di nomi di interfacce” a pagina 803
- “Assegnazione di nomi di componenti personalizzati” a pagina 803

## Linee guida generali per l’assegnazione di nomi

In questa sezione sono illustrate le linee guida relative all’assegnazione di nomi per la scrittura di codice ActionScript. Le convenzioni di denominazione sono importanti per la creazione di codice organizzato in modo logico. Lo scopo principale è migliorare la leggibilità del codice ActionScript 2.0. Tutte le variabili devono avere nomi univoci. In Flash Player 7 e versioni successive i nomi fanno distinzione tra maiuscole e minuscole. Non utilizzare lo stesso nome con maiuscole e minuscole diverse, perché si potrebbe generare confusione durante la lettura del codice da parte di altri programmatore e potrebbero insorgere problemi con le versioni precedenti di Flash, in cui la differenza tra maiuscole e minuscole non è imposta. Quando si assegnano nomi a elementi Flash, ad esempio variabili, file e classi, tenere presenti le seguenti linee guida.

- Limitare l’uso di abbreviazioni.

Usare le abbreviazioni in modo coerente. Un’abbreviazione deve essere associata sempre e in modo chiaro a un unico elemento. L’abbreviazione “sec”, ad esempio, può indicare “sezione” e “secondo”.

- Concatenare le parole per creare nomi.

In tal caso, utilizzare l’iniziale maiuscola per ogni parola per migliorare la leggibilità delle parole. Scrivere, ad esempio, `mioProgetto` anziché `mioprogetto`.

- Assegnare un nome a un file descrivendo il processo o l’elemento, ad esempio `aggiungiUtente`.

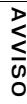
- Non utilizzare nomi non descrittivi per metodi o variabili.  
Se, ad esempio, si recuperano informazioni come il nome utente di un visitatore, utilizzare il metodo `getUserName()` anziché `getData()` che è meno descrittivo. Il nome utilizzato nell'esempio descrive cosa sta accadendo e non come viene ottenuto il risultato.
- Ridurre il più possibile la lunghezza dei nomi.  
Fare in modo che siano comunque descrittivi.

Nelle sezioni seguenti vengono forniti maggiori dettagli sull'assegnazione di nomi agli elementi del codice, ad esempio variabili, classi, pacchetti e costanti.

## Evitare parole riservate o costrutti del linguaggio

Quando si assegnano nomi a istanze e variabili, evitare l'uso di parole riservate che possono causare errori nel codice. Le parole riservate includono le *parole chiave* del linguaggio ActionScript.

Evitare inoltre di utilizzare qualsiasi parola nei linguaggi ActionScript 2.0 (ovvero *costrutto del linguaggio*) come nome di una variabile o di un'istanza. I costrutti ActionScript comprendono nomi di classi, classi di componenti, metodi, proprietà e interfacce.



Non utilizzare mai maiuscole e minuscole diverse, per evitare conflitti con le parole riservate. Un'istanza della classe `TextField` denominata, ad esempio, `textfield` non crea conflitti con `TextField`, perché Flash fa distinzione tra maiuscole e minuscole, ma non è conforme alle linee guida suggerite.

Nella tabella seguente sono elencate le parole riservate di ActionScript 2.0 che, se utilizzate come nomi di variabili, causano errori di script:

add	and	break	case
catch	class	continue	default
delete	do	dynamic	else
eq	extends	false	finally
for	function	ge	get
gt	if	ifFrameLoaded	implements
import	in	instanceof	interface
intrinsic	le	it	ne
new	not	null	on
onClipEvent	or	private	public
return	set	static	super

---

switch	tellTarget	this	throw
try	typeof	var	void
while	with		

---

Le seguenti parole sono riservate per uso futuro in Flash in base alla bozza della specifica del linguaggio ECMAScript (ECMA-262) Edizione 4. Non utilizzarle in quanto potrebbero essere adottate nelle future release di Flash.

---

as	abstract	Boolean	bytes
char	const	debugger	double
enum	export	final	float
goto	is	long	namespace
native	package	protected	short
synchronized	throws	transient	use
volatile			

---

## Assegnazione di nomi alle variabili

I nomi di variabili possono contenere solo lettere, numeri e segni di dollaro (\$). Non iniziare il nome di una variabile con un numero. Le variabili devono essere univoche. In Flash Player 7 e versioni successive viene fatta distinzione tra maiuscole e minuscole. Non utilizzare ad esempio i seguenti nomi:

```
my/warthog = true; // Contiene una barra
my warthogs = true; // Contiene uno spazio
my.warthogs = true; // Contiene un punto
5warthogs = 55; // Inizia con un numero
```

Se possibile, adottare la tipizzazione forte dei dati con le variabili, perché garantisce i seguenti vantaggi:

- Consente di utilizzare la funzionalità di completamento del codice per scrivere codice in modo più rapido.
- Genera errori nel pannello Output in modo da evitare un malfunzionamento senza alcuna segnalazione quando si compila un file SWF. Questi errori permettono di individuare e risolvere i problemi delle applicazioni.

Per tipizzare una variabile, definirla utilizzando la parola chiave var. Nell'esempio seguente, per la creazione di un oggetto LoadVars si utilizza la tipizzazione forte dei dati:

```
var paramsLv:LoadVars = new LoadVars();
```

La tipizzazione forte dei dati consente di avvalersi del completamento del codice, garantisce che il valore di `params.Lv` contenga un oggetto `LoadVars`, e assicura che l'oggetto `LoadVars` non venga utilizzato per memorizzare dati numerici o stringhe. Dato che la tipizzazione forte dei dati richiede che venga specificata la parola chiave `var`, non è possibile utilizzarla con variabili globali o proprietà all'interno di un oggetto o di un array. Per ulteriori informazioni sull'uso della tipizzazione forte dei dati con le variabili, vedere [“Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati” a pagina 344](#).

**NOTA**

La tipizzazione forte dei dati non rallenta le prestazioni di un file SWF. La verifica del tipo viene eseguita in fase di compilazione, ovvero quando viene creato il file SWF, e non in fase di runtime.

Se si assegnano nomi alle variabili del codice, adottare le linee guida seguenti:

- Tutte le variabili devono avere nomi univoci.
- Non utilizzare lo stesso nome di variabile con maiuscole e minuscole diverse.  
Non usare, ad esempio, `firstname` e `firstName` come variabili diverse in un'applicazione. Sebbene in Flash Player 7 e versioni successive per i nomi venga fatta distinzione tra maiuscole e minuscole, l'uso dello stesso nome di variabile con maiuscole e minuscole diverse potrebbe generare confusione durante la lettura del codice da parte di altri programmatori e creare problemi con le versioni precedenti di Flash, in cui la differenza tra maiuscole e minuscole non è imposta.
- Non utilizzare parole che fanno parte del linguaggio ActionScript 1.0 o 2.0 come nomi di variabili.  
In particolare, non usare parole chiave come nomi di istanza, poiché si verificheranno errori nel codice. Non fare affidamento sulla differenza tra maiuscole e minuscole per evitare conflitti e garantire il corretto funzionamento del codice.
- Non fare ricorso a variabili che fanno parte di costrutti di programmazione comuni.  
Non usare costrutti di linguaggio tipici di altri linguaggi di programmazione, anche se Flash non include o supporta tali costrutti di linguaggio. Non utilizzare, ad esempio, le seguenti parole chiave come variabili:  
`textfield = "myTextField";`  
`switch = true;`  
`new = "funk";`

- Aggiungere sempre le annotazioni di tipo al codice.

Dette anche “uso della tipizzazione forte dei dati con le variabili” o “tipizzazione forte delle variabili”, le annotazioni di tipo aggiunte alle variabili sono importanti per:

- Generare errori in fase di compilazione per evitare che l'applicazione non venga eseguita correttamente senza avvisi.
- Attivare il completamento del codice.
- Aiutare gli altri utenti a comprendere il codice.

Per informazioni su come aggiungere annotazioni di tipo, vedere [“Informazioni sull'assegnazione dei tipi di dati e la tipizzazione forte dei dati”](#) a pagina 344.

- Limitare il ricorso al tipo Object.

Le annotazioni di tipo devono essere precise per migliorare le prestazioni. Usare un tipo Object soltanto quando non c'è un'alternativa ragionevole.

- Ridurre il più possibile la lunghezza dei nomi delle variabili, assicurando al tempo stesso la massima chiarezza.

Accertarsi che i nomi delle variabili siano descrittivi, ma non eccedere e non utilizzare nomi eccessivamente lunghi e complessi.

- Utilizzare nomi di variabili a un solo carattere esclusivamente per ottimizzare i cicli.

Facoltativamente è possibile utilizzare variabili a carattere singolo come variabili temporanee, ad esempio `i`, `j`, `k`, `m` e `n`. Questi nomi di variabili a un solo carattere devono tuttavia essere utilizzati solo per indici di cicli brevi oppure se l'ottimizzazione delle prestazioni e la velocità sono aspetti critici. L'esempio seguente mostra questo utilizzo:

```
var fontArr:Array = TextField.getFontList();
fontArr.sort();
var i:Number;
for (i = 0; i<fontArr.length; i++) {
 trace(fontArr[i]);
}
```

- Iniziare i nomi delle variabili con una lettera minuscola.

I nomi che iniziano con una lettera maiuscola sono riservati per classi, interfacce e così via.

- Utilizzare maiuscole e minuscole miste per le parole concatenate.

Ad esempio, usare `myFont` anziché `myfont`.

- Non utilizzare acronimi e abbreviazioni.

Fanno eccezione quegli acronimi o quelle abbreviazioni che rappresentano il modo standard di utilizzare un termine (ad esempio HTML o CFM). Per gli acronimi utilizzati comunemente, la leggibilità viene migliorata dall'uso di maiuscole e minuscole miste, ad esempio `newHTMLParser` anziché `newHTMLParser`.

- Fare ricorso a coppie complementari quando si crea un set di nomi di variabili correlate.  
È possibile ad esempio utilizzare coppie complementari per indicare il punteggio massimo e minimo di un gioco, nel modo seguente:

```
var minScoreNum:Number = 10; // Punteggio minimo
var maxScoreNum:Number = 500; // Punteggio massimo
```

## Assegnazione di nomi alle costanti

Le costanti possono essere utilizzate quando è necessario fare riferimento a una proprietà il cui valore non cambia mai. In questo modo è più semplice trovare errori di ortografia nel codice, che sarebbero invece difficili da rilevare utilizzando caratteri letterali, ed è possibile modificare il valore in un solo punto.

Mentre i nomi delle variabili devono essere composti da lettere minuscole o da maiuscole e minuscole miste, seguire queste linee guida per assegnare i nomi alle costanti statiche (variabili che non cambiano):

- I nomi delle costanti devono essere espressi in lettere maiuscole.
- Separare le parole con un carattere di sottolineatura.

Nel frammento di codice ActionScript seguente è illustrato un esempio di queste linee guida:

```
var BASE_URL:String = "http://www.macromedia.com"; // Costante
var MAX_WIDTH:Number = 10; // Costante
```

Non scrivere direttamente costanti numeriche a meno che non si tratti di 1, 0 o -1, utilizzabili in un ciclo `for` come valori di contatore.

## Assegnazione di nomi alle variabili booleane

Iniziare le variabili booleane con la parola "is"; per sua natura, infatti, un valore booleano "is" (è) o "isn't" (non è). È pertanto possibile utilizzare quanto segue per stabilire se un neonato è una femmina o non lo è (un valore booleano):

`isGirl`

Ottimamente, per una variabile che indica se un utente ha effettuato il login (o no), è possibile utilizzare quanto segue:

`isLoggedIn`

## Assegnazione di nomi a funzioni e metodi

Se si assegnano nomi alle funzioni e ai metodi del codice, adottare le linee guida seguenti: Per informazioni su come scrivere funzioni e metodi, consultare il [Capitolo 5, “Funzioni e metodi”](#)

- Utilizzare nomi descrittivi.
- Utilizzare maiuscole e minuscole miste per le parole concatenate.  
Un esempio significativo è rappresentato da `singLoud()`.
- Iniziare i nomi delle funzioni e dei metodi con una lettera minuscola.
- Descrivere il valore che verrà restituito nel nome della funzione.  
Se, ad esempio, si restituisce il titolo di un brano, denominare la funzione `getBranoCorrente()`.
- Stabilire una regola di denominazione fissa per correlare funzioni simili.  
In ActionScript 2.0 non è infatti consentito l'overload. Nella programmazione orientata agli oggetti, per *overload* si intende la possibilità che le funzioni personalizzate si comportino in modo diverso a seconda del tipo di dati che viene loro passato.
- Assegnare ai metodi un nome con forma verbale.  
È possibile concatenare il nome, se contiene un verbo.. Per la maggior parte dei nomi di metodi si utilizzano verbi, poiché i metodi eseguono operazioni su oggetti.

Di seguito sono riportati alcuni esempi di nomi di metodi:

```
sing();
boogie();
singLoud();
danceFast();
```

## Assegnazione di nomi a classi e oggetti

Quando si crea un nuovo file di classe, utilizzare le linee guida seguenti per assegnare il nome alla classe e al file ActionScript. Di seguito sono riportati alcuni esempi di nomi di classe con la formattazione corretta:

```
class Widget;
class PlasticWidget;
class StreamingVideo;
```

In una classe possono essere presenti variabili pubbliche e private. La classe può contenere variabili che non devono essere impostate dagli utenti o a cui gli utenti non devono accedere direttamente. Impostare queste variabili come private e renderle accessibili agli utenti solo tramite metodi getter/setter.

Le linee guida seguenti si applicano all'assegnazione di nomi alle classi:

- Iniziare un nome di classe con la maiuscola
- I nomi delle classi devono avere l'iniziale maiuscola e iniziare con la lettera maiuscola quando è un composto o una parola concatenata.

Iniziare un nome di un composto o una parola concatenata con una lettera maiuscola. Un esempio significativo è rappresentato da `NuovoMembro`.
- I nomi di classe sono generalmente nomi o nomi qualificati.

Un qualificatore descrive il nome o l'espressione. Anziché utilizzare solo il termine "membro", ad esempio, è possibile qualificare il nome utilizzando `NuovoMembro` o `VecchioMembro`.
- Privilegiare la chiarezza rispetto alla brevità.
- Non utilizzare acronimi e abbreviazioni.

Fanno eccezione quegli acronimi o quelle abbreviazioni che rappresentano il modo standard di utilizzare un termine (ad esempio HTML o CFM). Per gli acronimi utilizzati comunemente, la leggibilità viene migliorata dall'uso di maiuscole e minuscole miste, ad esempio `NewHtmlParser` anziché `NewHTMLParser`.
- Utilizzare nomi semplici e significativi che siano descrittivi del contenuto della classe.

Per evitare di essere vaghi o fuorvianti, utilizzare nomi generici.
- A volte il nome di una classe è una parola composta.

Un qualificatore può descrivere il nome o l'espressione. Anziché utilizzare solo il termine "membro", ad esempio, è possibile qualificare il nome utilizzando `NuovoMembro` o `VecchioMembro`.
- Non utilizzare parole al plurale nel nome della classe (come `Streghe` o `PiratiCalvi`).

Nella maggior parte dei casi, è consigliabile lasciare le parole come nomi non qualificati. Un qualificatore descrive il nome o l'espressione. Anziché utilizzare solo il termine "gatto" o "bucaniere", ad esempio, è possibile qualificare il nome utilizzando `GattoNero` o `VecchioBucaniere`.
- Non utilizzare il nome della classe nella proprietà della stessa perché si creerebbe una ridondanza.

Non avrebbe senso, ad esempio, creare un nome come `Gatto.BaffiGatto`. Sarebbe sufficiente e più chiaro `Gatto.Baffi`.
- Non utilizzare nomi che possono essere interpretati come verbi.

Ad esempio, `Potere` o `Calzare`. Potrebbero generare confusione con i metodi, gli stati o altre attività dell'applicazione.
- Usare nomi di classe univoci per ogni classe in una applicazione.

- Non assegnare i nomi delle classi in modo che possano essere in conflitto con i nomi delle classi incorporate in Flash.
- Cercare di esplicitare la relazione tra una classe e la relativa gerarchia.  
È utile per visualizzare la relazione di una classe all'interno di un'applicazione.  
L'implementazione dell'interfaccia Computer potrebbe essere, ad esempio,  
ComputerDesktop, ComputerLaptop e ComputerPortatile.

Per informazioni sulle interfacce, vedere [Capitolo 8, “Interfacce”](#).

## Assegnazione di nomi ai pacchetti

Per i nomi dei pacchetti viene comunemente utilizzata la convenzione di denominazione “dominio invertito”. Esempi di nomi di dominio invertito includono com.macromedia per macromedia.com e org.dominio per dominio.org.

Adottare le linee guida seguenti per l'assegnazione dei nomi ai pacchetti:

- Il prefisso nei nomi di pacchetto deve avere lettere minuscole  
Ad esempio, com, mx o org.
- Inserire le classi correlate (classi con funzionalità correlate) nello stesso pacchetto.
- Iniziare i nomi di pacchetto con un prefisso coerente.  
Ad esempio, è possibile utilizzare com.macromedia.projectName per garantire la coerenza. Un ulteriore esempio è rappresentato da com.macromedia.docs.learnAS2.Users per il manuale *Apprendimento di ActionScript 2.0 in Flash*.
- Utilizzare un nome di pacchetto chiaro e descrittivo.  
È importante che illustri le operazioni eseguite dal pacchetto. Nel caso, ad esempio, di un pacchetto denominato Pentagoni con cui vengono disegnati diversi tipi di pentagoni tramite l'API di disegno di Flash, il nome potrebbe essere com.macromedia.Pentagoni.
- Utilizzare maiuscole e minuscole miste per i nomi di pacchetti composti o concatenati.  
packageName rappresenta un esempio di nome di pacchetto concatenato. Utilizzare lettere minuscole per il prefisso nei nomi di pacchetto (com, org e così via).
- Non utilizzare caratteri di sottolineatura o simboli di dollaro.

## Assegnazione di nomi di interfacce

Iniziare i nomi di interfaccia con una “I” maiuscola facilita la distinzione tra interfacce e classi. Nel seguente nome di interfaccia, `IEmployeeRecords`, viene utilizzata l'iniziale maiuscola sia per la prima parola che per le parole concatenate, nel modo seguente:

```
interface IEmployeeRecords{}
```

Vengono inoltre applicate le convenzioni seguenti:

- I nomi di interfaccia hanno la prima lettera maiuscola.  
La stessa notazione dei nomi di classe.
- I nomi di interfaccia sono rappresentati in genere da aggettivi.  
`Stampabile` ne è un esempio.

Per ulteriori informazioni sulle interfacce, consultare il [Capitolo 8, “Interfacce”](#)

## Assegnazione di nomi di componenti personalizzati

Nei nomi dei componenti la prima lettera e le lettere di ogni parola concatenata sono maiuscole. Nei seguenti componenti predefiniti dell'interfaccia utente, ad esempio, vengono utilizzate parole concatenate e ogni parola ha l'iniziale maiuscola:

- CheckBox
- ComboBox
- DataGridView
- DateChooser
- TextField
- MenuBar
- NumericStepper
- ProgressBar
- RadioButton
- ScrollPane
- TextArea
- TextInput

I nomi di componenti in cui non vengono utilizzate parole concatenate iniziano con la lettera maiuscola.

Se si sviluppano componenti personalizzati, adottare una convenzione per l'assegnazione dei nomi per evitare incompatibilità con i nomi dei componenti Macromedia. I nomi dei componenti personalizzati devono essere diversi da quelli dei componenti predefiniti inclusi in Flash. L'adozione di una convenzione coerente per l'assegnazione dei nomi contribuisce alla prevenzione di conflitti tra i nomi.

Tenere a mente che le convenzioni per l'assegnazione dei nomi adottate in questa sezione sono solo linee guida. L'aspetto più importante rimane sempre l'utilizzo di uno schema di assegnazione dei nomi che funzioni per il contesto in cui viene adottato e l'utilizzo coerente dello stesso.

## Uso di commenti nel codice

In questa sezione viene descritto come utilizzare i commenti nel codice. I commenti documentano le decisioni prese nel codice e devono spiegare *come* e *perché* è stata eseguita una determinata scelta nel codice. Potrebbero descrivere, ad esempio, una soluzione alternativa. In questo modo è più facile per altri sviluppatori trovare il codice correlato da aggiornare o correggere. Il problema potrebbe essere risolto, infine, in una versione futura di Flash o Flash Player; nel qual caso la soluzione alternativa non sarebbe più necessaria.

Per ulteriori informazioni sulla scrittura di commenti nel codice ActionScript, consultare le sezioni seguenti:

- “[Scrittura di commenti significativi](#)” a pagina 804
- “[Aggiunta di commenti alle classi](#)” a pagina 806

### Scrittura di commenti significativi

L'uso coerente di commenti nel codice ActionScript 2.0 consente di descrivere aree di codice complesse o interazioni importanti, che non sarebbero altrimenti di facile comprensione.

Devono descrivere in modo chiaro lo scopo del codice e non limitarsi a tradurre il codice. Se un'operazione non risulta immediata leggendo il codice, aggiungere commenti per illustrarla.

Se si utilizza lo strumento Formattazione automatica con il codice, i commenti finali (vedere “[Commenti finali](#)” a pagina 98) vengono spostati alla riga successiva. È possibile aggiungere i commenti dopo avere formattato il codice, se non si desidera che vengano spostati quando si utilizza lo strumento Formattazione automatica.

Per informazioni sull'utilizzo di commenti nelle classi, vedere “[Aggiunta di commenti alle classi](#)” a pagina 806.

Se si aggiungono commenti al codice, adottare le linee guida seguenti:

- Utilizzare commenti a blocchi (`/* e */`) per commenti su più righe e commenti a riga singola per commenti di lunghezza limitata.  
È inoltre possibile utilizzare un *commento finale* sulla stessa riga del codice ActionScript, se necessario.
- Non utilizzare commenti per tradurre il codice ActionScript.  
È superfluo commentare elementi del codice ActionScript che sono già chiari.
- Inserire commenti per gli elementi che non risultano chiari nel codice.  
In particolare, aggiungere commenti quando il soggetto non è descritto nei paragrafi circostanti.
- Non aggiungere commenti disordinati.  
Una riga di commenti disordinati contiene spesso segni uguale (=) o asterischi (\*). Utilizzare invece spazi vuoti per separare i commenti dal codice ActionScript.

**NOTA**

Se il codice ActionScript viene formattato tramite lo strumento Formattazione automatica, lo spazio vuoto viene rimosso. Ricordarsi quindi di aggiungerlo nuovamente oppure di utilizzare righe di commenti singole (/) per mantenere la spaziatura. Sarà semplice rimuovere queste righe dopo la formattazione del codice.

- Prima di distribuire il progetto, rimuovere eventuali commenti superflui.  
Se nel codice ActionScript sono presenti troppi commenti, provare a riscrivere parte del codice. Se si ritiene di dover inserire molti commenti sul funzionamento del codice ActionScript, in genere significa che il codice non è scritto in modo chiaro e secondo le convenzioni illustrate.

**NOTA**

L'uso di commenti nel codice ActionScript è particolarmente importante se si utilizza il codice a scopo didattico. Aggiungere commenti al codice, ad esempio, se si creano applicazioni esemplificative per insegnare l'uso di Flash oppure se si scrivono esercitazioni sul codice ActionScript.

## Aggiunta di commenti alle classi

In un file di interfaccia o di classe tipico sono presenti due tipi di commenti: *commenti di documentazione* e *commenti di implementazione*.

**NOTA**

I commenti di documentazione e di implementazione non sono rappresentati formalmente nel linguaggio ActionScript. Tuttavia, sono comunemente utilizzati dagli sviluppatori durante la scrittura di file di classe e di interfaccia.

I commenti di documentazione vengono utilizzati per descrivere le specifiche del codice, ma non l'implementazione. I commenti di implementazione, invece, vengono utilizzati per impostare porzioni di codice come commento o per inserire commenti sull'implementazione di determinate sezioni di codice. i commenti di documentazione sono delimitati con `/**` e `*/`, mentre i commenti di implementazione sono delimitati con `/*` e `*/`.

Possono descrivere interfacce, classi, metodi e funzioni di costruzione. Includere un unico commento di documentazione per classe, interfaccia o membro, inserendolo subito prima della dichiarazione. Se si desidera inserire ulteriori informazioni di documentazione e lo spazio nei commenti di documentazione non è sufficiente, fare ricorso ai commenti di implementazione, utilizzando il formato dei commenti a blocchi o a riga singola.

Iniziare le classi con un commento standard, che utilizza il formato seguente:

```
/**
 * Classe User
 * Versione 1.2
 * 3/21/2004
 * Copyright Macromedia, Inc.
 */
```

Dopo i commenti di documentazione, dichiarare la classe. I commenti di implementazione dovrebbero essere inseriti subito dopo la dichiarazione.

**NOTA**

Non inserire commenti che non si riferiscano direttamente alla classe letta, ad esempio commenti che descrivono il pacchetto corrispondente.

Utilizzare commenti a blocchi, a riga singola e finali all'interno del corpo della classe per inserire commenti sul codice ActionScript. Per ulteriori informazioni sull'uso dei commenti nei file di classe, vedere “[Aggiunta di commenti alle classi](#)” a pagina 806.

# Convenzioni di codifica ActionScript

Uno degli aspetti fondamentali della programmazione è la coerenza, in relazione sia agli schemi di assegnazione dei nomi delle variabili (trattati in “[Convenzioni di denominazione](#)” a pagina 793), sia alla formattazione del codice (trattata in “[Formattazione della sintassi ActionScript](#)” a pagina 825), sia agli standard per la scrittura di codice e alla posizione di inserimento del codice ActionScript 2.0, che è trattato in questa sezione. Se il codice è organizzato in modo appropriato ed è conforme agli standard, il debug e l'aggiornamento del codice risultano estremamente semplificati.

Per ulteriori informazioni sulle convenzioni di codifica, consultare gli argomenti seguenti:

- “[Mantenere il codice ActionScript in una sola posizione](#)” a pagina 807
- “[Associazione di codice agli oggetti](#)” a pagina 808
- “[Gestione dell'area di validità](#)” a pagina 809
- “[Strutturazione di un file di classe](#)” a pagina 813
- “[Informazioni sull'uso delle funzioni](#)” a pagina 821

## Mantenere il codice ActionScript in una sola posizione

Quando possibile, inserire il codice ActionScript 2.0 in una sola posizione, ad esempio in uno o più file ActionScript esterni o nel fotogramma 1 della linea temporale (quando è posizionato nella linea temporale viene detto *script di fotogramma*).

Se il codice ActionScript viene inserito in un script di fotogramma, aggiungerlo al primo o al secondo fotogramma della linea temporale, in un livello denominato *Azioni*, che corrisponde al primo o il secondo livello della linea temporale. A volte può essere necessario creare due livelli per separare le funzioni di ActionScript. Con alcune applicazioni Flash il codice non si trova sempre in una sola posizione, in particolare se si utilizzano schermate o comportamenti. Fatta eccezione per queste rare eccezioni, tutto il codice può solitamente essere inserito in una sola posizione. Di seguito sono elencati i vantaggi offerti dall'inserimento del codice ActionScript in una sola posizione:

- Il codice può essere individuato in modo semplice anche in file sorgente potenzialmente complessi.
- Il debug del codice risulta semplificato.

Uno degli aspetti più ostici del debug di un file FLA è l'individuazione di tutto il codice. Dopo avere individuato tutto il codice, è necessario comprendere l'interazione tra le diverse porzioni di codice insieme al file FLA. Se tutto il codice viene aggiunto a un solo fotogramma, è più facile eseguirne il debug perché si trova in una posizione centralizzata, e questo tipo di problemi è meno frequente. Per informazioni sull'associazione di codice agli oggetti e sulla decentralizzazione del codice, vedere “[Associazione di codice agli oggetti](#)” a pagina 808. Per informazioni sui comportamenti e il codice decentralizzato, vedere Capitolo 3, “[Procedure ottimali per l'uso dei comportamenti](#)” in *Uso di Flash*.

## Associazione di codice agli oggetti

È necessario evitare di associare codice ActionScript agli oggetti (ad esempio istanze di pulsanti o clip filmato) in un file FLA, anche se in applicazioni prototipo o semplici. Per associare codice a un oggetto è necessario selezionare un clip filmato, un componente o un'istanza di pulsante, aprire l'Editor di ActionScript (il pannello Azioni o la finestra Script) e aggiungere il codice ActionScript tramite le funzioni di gestione `on()` o `onClipEvent()`.

Si sconsiglia di adottare questa procedura per i seguenti motivi:

- È difficile individuare codice ActionScript associato a oggetti e in questo caso è difficile modificare i file FLA.
- È difficile eseguire il debug di codice ActionScript associato a oggetti.
- È più pratico aggiornare codice ActionScript scritto sulla linea temporale o in classi.
- Lo stile del codice ActionScript associato agli oggetti non è conforme alle linee guida suggerite.
- Il codice ActionScript associato a oggetti obbliga gli studenti e gli utenti che lo leggono ad apprendere sintassi aggiuntiva e stili diversi per la scrittura del codice, che sono spesso limitati e poco professionali.
- Generalmente gli utenti devono apprendere di nuovo come scrivere le funzioni e così via su una linea temporale.

Si potrebbe pensare che sia più semplice imparare a scrivere in linguaggio ActionScript se il codice viene associato a un oggetto e, inoltre, che l'aggiunta di codice semplice o la scrittura o l'insegnamento di ActionScript sia più facile in questo modo. Tuttavia, la differenza tra i due stili (codice inserito negli oggetti e script di fotogrammi) potrebbe invece generare confusione negli sviluppatori che stanno apprendendo ActionScript e si dovrebbe evitare. Inoltre, gli utenti che apprendono come scrivere codice associato a oggetti spesso devono apprendere di nuovo come inserire il codice equivalente come script di fotogramma. Per questo motivo la coerenza durante tutto il processo di apprendimento, ad esempio su come scrivere script di fotogramma, ha notevoli vantaggi.

L'associazione di un codice ActionScript a un pulsante myBtn ha l'aspetto seguente. Evitare questo metodo:

```
on (release) {
 // Esegue un'azione.
}
```

Tuttavia, l'inserimento del codice ActionScript equivalente su una linea temporale ha l'aspetto seguente:

```
// codice corretto
myBtn.onRelease = function() {
 // Esegue un'azione.
};
```

Per ulteriori informazioni sulla sintassi di ActionScript, vedere [“Formattazione della sintassi ActionScript” a pagina 825.](#)

**NOTA**

Se si fa ricorso ai comportamenti e alle schermate, le procedure sono diverse e a volte si rende necessario associare codice agli oggetti. Per ulteriori informazioni, vedere [Capitolo 3, “Procedure ottimali per l'uso dei comportamenti” in Uso di Flash.](#)

## Gestione dell'area di validità

Per area di validità si intende l'ambito in cui la variabile viene riconosciuta e può essere utilizzata in un file SWF, ad esempio sulla linea temporale, in tutta l'applicazione o a livello locale in una funzione. Quando si scrive il codice, i modi per creare un riferimento a un'area di validità sono solitamente più d'uno. Per l'uso corretto delle aree di validità è possibile creare codice ActionScript portatile e riutilizzabile, senza correre il rischio di pregiudicare le applicazioni quando si creano nuovi moduli.

È importante comprendere la differenza tra le aree di validità \_root e \_global. L'area di validità \_root è univoca per ogni file SWF caricato. L'area di validità globale è applicabile a tutte le linee temporali e tutte le aree di validità all'interno dei file SWF. Fare ricorso all'indirizzamento relativo anziché a riferimenti alle linee temporali \_root per garantire la riutilizzabilità e la portabilità del codice. Per ulteriori informazioni sulla gestione dell'area di validità, consultare le seguenti sezioni:

[“Informazioni sulle variabili e l'area di validità” a pagina 362](#)

[“Informazioni su area di validità e identificazione” a pagina 86](#)

[“Nozioni fondamentali sulle classi e l'area di validità” a pagina 260.](#)

## Non utilizzare target assoluti (\_root)

Per evitare di utilizzare `_root` è possibile fare riferimento alle istanze in diversi modi che vengono presentati più avanti in questa sezione. Non utilizzare `_root` in ActionScript 2.0 perché i file SWF che vengono caricati in altri file SWF potrebbero non funzionare correttamente. L'identificatore `_root` fa riferimento al file SWF in corso di caricamento e non al file SWF che utilizza l'indirizzamento relativo anziché `_root`. Questo problema limita la portabilità del codice nei file SWF che vengono caricati in un altro file e, in particolare, in componenti e clip filmato. Per risolvere i problemi è possibile utilizzare `_lockroot`, ma solo quando necessario, ad esempio se si carica un file SWF ma non si ha accesso al file FLA. Per ulteriori informazioni sull'uso di `_lockroot`, vedere “[Uso di `\_lockroot`](#)” a pagina 810.

Utilizzare le parole chiave `this`, `this._parent` o `_parent` anziché `_root`, in base alla posizione del codice ActionScript 2.0. Nell'esempio seguente è illustrato l'indirizzamento relativo:

```
myClip.onRelease = function() {
 trace(this._parent.myButton._x);
};
```

Per tutte le variabili deve essere specificata l'area di validità, ad eccezione delle variabili che rappresentano parametri di funzioni e delle variabili locali. L'area di validità delle variabili deve basarsi sul percorso corrente della variabile, se possibile, facendo ricorso all'indirizzamento relativo, ad esempio alla proprietà `this`. Per ulteriori informazioni sull'utilizzo della proprietà `this`, vedere `this` property nella *Guida di riferimento di ActionScript 2.0*.

## Uso di `_lockroot`

È possibile utilizzare `_lockroot` per specificare il contenuto è un modo per risolvere i problemi legati all'area di validità generati a volte dall'uso non appropriato di `_root`. Sebbene consenta di risolvere molti problemi che si verificano con le applicazioni, l'uso di `_lockroot` deve essere considerato semplicemente una soluzione alternativa per i problemi causati dall'uso di `_root`. In caso di problemi con il caricamento del contenuto in un file SWF o in un'istanza di un componente, provare ad applicare `_lockroot` a un clip filmato che carica il contenuto. Se si dispone, ad esempio, di un clip filmato denominato `myClip` che carica contenuto, e questo clip filmato non funziona più subito dopo il caricamento, provare a utilizzare il codice seguente, che viene inserito su una linea temporale:

```
this._lockroot = true;
```

## Uso della parola chiave this

Se possibile, utilizzare la parola chiave `this` come prefisso, indipendentemente dal fatto che il codice funzioni anche se viene omessa. Utilizzare la parola chiave `this` per comprendere se un metodo o una proprietà appartiene a una determinata classe. Per una funzione sulla linea temporale, ad esempio, utilizzare codice ActionScript 2.0 con il formato seguente:

```
circleClip.onPress = function() {
 this.startDrag();
};

circleClip.onRelease = function() {
 this.stopDrag();
};
```

Per una classe, utilizzare il formato seguente:

```
class User {
 private var username:String;
 private var password:String;
 function User(username:String, password:String) {
 this.username = username;
 this.password = password;
 }
 public function get username():String {
 return this.username;
 }
 public function set username(username:String):Void {
 this.username = username;
 }
}
```

Se la parola chiave `this` viene aggiunta in modo coerente in queste situazioni, è più semplice leggere e comprendere il codice ActionScript 2.0.

## Informazioni sull'area di validità nelle classi

Se si trasferisce un codice nelle classi ActionScript 2.0, potrebbe essere necessario modificare il tipo di uso della parola chiave `this`. Se, ad esempio, è presente un metodo di classe che utilizza una funzione di callback (quale il metodo `onLoad` della classe `LoadVars`), può risultare difficile comprendere se la parola chiave `this` si riferisce alla classe o all'oggetto `LoadVars`. In questo caso, potrebbe essere necessario creare un puntatore alla classe corrente, come illustrato nell'esempio seguente:

```
class Product {
 private var m_products_xml:XML;
 // Funzione di costruzione
 // targetXmlStr contiene il percorso a un file XML
 function Product(targetXmlStr:String) {
 /* Crea un riferimento locale alla classe corrente.
```

```

Anche all'interno del gestore di eventi onLoad di XML
è possibile fare riferimento alla classe corrente anziché solo al
pacchetto XML. */
var thisObj:Product = this;
// Crea una variabile locale che viene utilizzata per caricare il file
XML.
var prodXml:XML = new XML();
prodXml.ignoreWhite = true;
prodXml.onLoad = function(success:Boolean) {
 if (success) {
 /* Se il file XML viene caricato e analizzato correttamente,
 impostare la variabile m_products_xml della classe sul documento
 XML analizzato e chiamare la funzione init. */
 thisObj.m_products_xml = this;
 thisObj.init();
 } else {
 /* Si è verificato un errore durante il caricamento del file XML.
 */
 trace("error loading XML");
 }
};
// Inizia a caricare il documento XML
prodXml.load(targetXmlStr);
}
public function init():Void {
 // Visualizza il pacchetto XML
 trace(this.m_products_xml);
}
}

```

Dato che si tenta di fare riferimento alla variabile di un membro privato all'interno di un gestore onLoad, la parola chiave `this` si riferisce all'istanza `prod_Xml` e non alla classe `Product`, come si potrebbe supporre. Per questo motivo, è necessario creare un puntatore al file della classe locale affinché sia possibile fare riferimento direttamente alla classe dal gestore onLoad.

Per ulteriori informazioni sulle classi, consultare il “[Nozioni fondamentali sulle classi e l'area di validità](#)” a pagina 260. Per ulteriori informazioni sull'area di validità, vedere “[Gestione dell'area di validità](#)” a pagina 809.

## Strutturazione di un file di classe

Le classi vengono create in file ActionScript 2.0 separati che vengono importati in un file SWF durante la compilazione.

Le classi vengono create in file ActionScript 2.0 separati che vengono importati in un file SWF durante la compilazione di un'applicazione. Per creare un file di classe, il codice deve essere scritto con una determinata metodologia e un certo ordine, come illustrato nelle sezioni seguenti:

Le convenzioni per strutturare un file di classe illustrate di seguito indicano come devono essere ordinate le parti di una classe per aumentare l'efficienza e migliorare la leggibilità del codice.

### Per strutturare un file di classe, utilizzare gli elementi seguenti:

1. Aggiungere i commenti di documentazione che comprendono una descrizione generale del codice e informazioni sull'autore e sulla versione.
2. Aggiungere le istruzioni di importazione (se necessarie).
3. Scrivere una dichiarazione di classe o una dichiarazione d'interfaccia, ad esempio:

```
UserClass{...}
```

4. Includere gli eventuali commenti relativi all'implementazione della classe o dell'interfaccia, aggiungendo informazioni relative all'intera classe o all'intera interfaccia.
5. Aggiungere tutte le variabili statiche.

Scrivere prima le variabili di classe statiche, quindi le variabili di classe private.

6. Aggiungere le variabili dell'istanza.

Scrivere prima le variabili dei membri pubblici, quindi quelle dei membri privati.

7. Aggiungere l'istruzione per la funzione di costruzione, come nell'esempio seguente:

```
public function UserClass(username:String, password:String) {...}
```

8. Scrivere i metodi,

raggruppandoli per funzionalità e non per accessibilità o area di validità. Questo tipo di organizzazione dei metodi consente di migliorare la leggibilità e la chiarezza del codice.

9. Scrivere i metodi getter/setter nel file di classe.

## Linee guida per creare una classe

Quando si crea un file di classe, tenere presenti le linee guida seguenti:

- Inserire una sola dichiarazione per riga.
- Non inserire più dichiarazioni sulla stessa riga.

Formattare le dichiarazioni come illustrato nell'esempio seguente:

```
var prodSkuNum:Number; // Numero di identificazione del prodotto (SKU)
var prodQuantityNum:Number; // Quantità di prodotti
```

In questo esempio il formato è migliore perché le dichiarazioni non sono state inserite sulla stessa riga. Inserire le dichiarazioni all'inizio di un blocco di codice.

- Inizializzare le variabili locali quando vengono dichiarate.  
Le proprietà di una classe devono essere inizializzate nella dichiarazione soltanto se l'inizializzatore è una costante compile time.
- Dichiare le variabili prima di utilizzarle.  
Sono inclusi i cicli.
- Non utilizzare dichiarazioni locali che nascondono dichiarazioni di livello superiore.

Ad esempio non dichiarare una variabile due volte, come illustrato nell'esempio seguente:

```
var counterNum:Number = 0;
function myMethod() {
 for (var counterNum:Number = 0; counterNum<=4; counterNum++) {
 // Istruzioni;
 }
}
```

Il codice dichiara la stessa variabile all'interno di un blocco interno. Evitare questo tipo di sintassi.

- Non assegnare molte variabili a un solo valore in un'istruzione.  
Se non si segue questa convenzione, il codice risulta difficile da leggere, come illustrato nel codice ActionScript seguente:  

```
playBtn.onRelease = playBtn.onRollOut = playsound;
```

  
o  

```
class User {
 private var m_username:String, m_password:String;
}
```
- Rendere pubblico un metodo o una proprietà soltanto se deve essere pubblico per qualche motivo. In caso contrario, rendere privati i metodi e le proprietà.

- *Limitare* l'uso delle funzioni getter/setter nel file di classe.  
Le funzioni getter/setter sono eccellenti per molti scopi (vedere “[Informazioni sui metodi getter e setter](#)” a pagina 229), tuttavia un uso eccessivo può indicare la possibilità di migliorare l'architettura o l'organizzazione di un'applicazione.
- Impostare la maggior parte delle variabili di membro come private, a meno che non esista un motivo per renderle pubbliche.  
Dal punto di vista della progettazione è decisamente preferibile impostare le variabili di membro come private e consentirvi l'accesso solo tramite un gruppo di funzioni getter/setter.

## Uso del prefisso this nei file di classe

Utilizzare la parola chiave `this` come prefisso all'interno delle classi per i metodi e le variabili di membro. Sebbene non sia necessario, l'uso di un prefisso facilita la comprensione dell'appartenenza di una proprietà o un metodo a una classe. In caso contrario, non è possibile determinare se la proprietà o il metodo appartiene alla superclasse.

È inoltre possibile utilizzare come prefisso il nome di una classe per le variabili statiche e i metodi anche all'interno di una classe per qualificare in modo migliore i riferimenti e rendere il codice più leggibile. A seconda dell'ambiente di scrittura del codice in uso, i prefissi potrebbero consentire il completamento del codice e i suggerimenti sul codice. Il codice seguente dimostra come fare precedere una proprietà statica con un nome di classe:

```
class Widget {
 public static var widgetCount:Number = 0;
 public function Widget() {
 Widget.widgetCount++;
 }
}
```

**NOTA**

L'aggiunta dei prefissi non è obbligatoria e molti sviluppatori non la ritengono necessaria, tuttavia Macromedia consiglia di aggiungere la parola chiave `this` come prefisso, perché permette di migliorare la leggibilità e di scrivere codice chiaro mediante l'indicazione del contesto.

## Informazioni sull'inizializzazione

Come valore iniziale delle variabili, assegnare un valore predefinito o consentire il valore `undefined`, come illustrato nell'esempio di classe seguente. Quando si inizializzano proprietà inline, l'espressione a destra di un'assegnazione deve essere una costante della fase di compilazione, vale a dire che l'espressione non può indicare alcun dato impostato o definito nella fase di runtime. Le costanti di compilazione includono caratteri letterali di stringa, numeri, valori booleani, `null` e `undefined`, oltre alle funzioni di costruzione per le seguenti classi di primo livello: `Array`, `Boolean`, `Number`, `Object` e `String`. Questa classe imposta i valori iniziali di `m_username` e `m_password` su stringhe vuote:

```
class User {
 private var m_username:String = "";
 private var m_password:String = "";
 function User(username:String, password:String) {
 this.m_username = username;
 this.m_password = password;
 }
}
```

Eliminare le variabili o impostarle su `null` quando non sono più necessarie. L'impostazione delle variabili su `null` può consentire di migliorare le prestazioni. Questo processo viene detto *garbage collection* (letteralmente, raccolta dei rifiuti). L'eliminazione delle variabili aiuta a ottimizzare l'uso della memoria in fase di runtime perché le risorse non più necessarie vengono rimosse dal file SWF. È preferibile eliminare le variabili piuttosto che impostarle su `null`. Per ulteriori informazioni sulle prestazioni, vedere [“Ottimizzazione del codice” a pagina 824](#).



Flash Player 8 ha introdotto miglioramenti nel processo di garbage collection in Flash Player.

Per ulteriori informazioni sull'assegnazione di nomi alle variabili, vedere [“Assegnazione di nomi alle variabili” a pagina 796](#). Per ulteriori informazioni sull'eliminazione di oggetti, vedere `delete` statement nella *Guida di riferimento di ActionScript 2.0*.

Se si utilizza ActionScript 2.0, uno dei modi più semplici per inizializzare il codice è rappresentato dall'uso delle classi. È possibile incorporare tutta l'inizializzazione di un'istanza all'interno della funzione di costruzione della classe oppure eseguirla tramite un metodo separato che viene chiamato in modo esplicito dopo la creazione della variabile, come illustrato nel codice seguente:

```
class Product {
 function Product() {
 var prodXml:XML = new XML();
 prodXml.ignoreWhite = true;
 prodXml.onLoad = function(success:Boolean) {
```

```

 if (success) {
 trace("loaded");
 } else {
 trace("error loading XML");
 }
 };
 prodXml.load("products.xml");
}
}

```

Il codice seguente potrebbe rappresentare la prima chiamata di funzione nell'applicazione e l'unica chiamata che viene eseguita per l'inizializzazione. Per il fotogramma 1 di un file FLA che carica codice XML potrebbe essere utilizzato codice ActionScript simile al seguente:

```

if (init == undefined) {
 var prodXml:XML = new XML();
 prodXml.ignoreWhite = true;
 prodXml.onLoad = function(success:Boolean) {
 if (success) {
 trace("loaded");
 } else {
 trace("error loading XML");
 }
 };
 prodXml.load("products.xml");
 init = true;
}

```

## Uso di istruzioni trace

Le istruzioni `trace` nei documenti facilitano il debug del codice durante la creazione di un file FLA. Utilizzando, ad esempio, un'istruzione `trace` e un ciclo `for`, è possibile visualizzare i valori delle variabili nel pannello Output, quali stringhe, array e oggetti, come illustrato nel seguente esempio:

```

var dayArr:Array = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"];
var numOfDays:Number = dayArr.length;
for (var i = 0; i<numOfDays; i++) {
 trace(i+": "+dayArr[i]);
}

```

Nel pannello Output vengono visualizzate le informazioni seguenti:

```

0: sun
1: mon
2: tue
3: wed
4: thu
5: fri
6: sat

```

L'uso di un'istruzione `trace` rappresenta un modo efficiente per eseguire il debug del codice ActionScript 2.0.

Quando si pubblica un file SWF è possibile rimuovere le istruzioni `trace` per migliorare lievemente le prestazioni di riproduzione. Prima di pubblicare un file SWF, aprire Impostazioni pubblicazione e selezionare Ometti azioni trace nella scheda Flash. Per ulteriori informazioni sull'utilizzo di una taccia, vedere `trace function` nella *Guida di riferimento di ActionScript 2.0*.

Anche lo strumento Debugger è utile per il debugging del codice ActionScript. Per ulteriori informazioni, vedere [Capitolo 18, “Esecuzione del debug delle applicazioni”](#).

## Informazioni sul prefisso super

Se si fa riferimento a un metodo nella classe principale, anteporre al metodo il prefisso `super` affinché altri sviluppatori possano capire da dove viene richiamato. Il frammento di codice ActionScript 2.0 seguente illustra l'assegnazione di un'area di validità corretta mediante il prefisso `super`:

Nell'esempio seguente, si creano due classi. Si utilizza la parola chiave `super` nella classe `Socks` per chiamare le funzioni della classe principale (`Clothes`). Sebbene le classi `Socks` e `Clothes` dispongano di un metodo denominato `getColor()`, usando `super` è possibile fare specificamente riferimento ai metodi e alle proprietà della classe di base. Creare un nuovo file AS denominato `Clothes.as` e immettere il codice seguente:

```
class Clothes {
 private var color:String;
 function Clothes(paramColor) {
 this.color = paramColor;
 trace("[Clothes] I am the constructor");
 }
 function getColor():String {
 trace("[Clothes] I am getColor");
 return this.color;
 }
 function setColor(paramColor:String):Void {
 this.color = paramColor;
 trace("[Clothes] I am setColor");
 }
}
```

Creare una nuova classe denominata Socks che estende la classe Clothes, come nell'esempio seguente:

```
class Socks extends Clothes {
 private var color:String;
 function Socks(paramColor:String) {
 this.color = paramColor;
 trace("[Socks] I am the constructor");
 }
 function getColor():String {
 trace("[Socks] I am getColor");
 return super.getColor();
 }
 function setColor(paramColor:String):Void {
 this.color = paramColor;
 trace("[Socks] I am setColor");
 }
}
```

Quindi creare un nuovo file AS o FLA e inserire il seguente codice ActionScript nel documento:

```
import Socks;
var mySock:Socks = new Socks("maroon");
trace(" -> "+mySock.getColor());
mySock.setColor("Orange");
trace(" -> "+mySock.getColor());
```

Il risultato seguente viene visualizzato nel pannello Output:

```
[Clothes] I am the constructor
[Socks] I am the constructor
[Socks] I am getColor
[Clothes] I am getColor
-> maroon
[Socks] I am setColor
[Socks] I am getColor
[Clothes] I am getColor
-> Orange
```

Se ci si scorda di inserire la parola chiave `super` nel metodo `getColor()` della classe `Socks`, il metodo `getColor()` può chiamare se stesso ripetutamente, provocando un errore nello script a causa di problemi di ricorsività infinita. Se non si utilizza la parola chiave `super`, il pannello Output visualizza il seguente errore:

```
[Socks] I am getColor
[Socks] I am getColor
...
[Socks] I am getColor
256 levels of recursion were exceeded in one action list.
This is probably an infinite loop.
Further execution of actions has been disabled in this SWF file.
```

## Evitare l'istruzione with

Uno dei concetti che possono risultare poco chiari per chi impara a scrivere in linguaggio ActionScript 2.0 è l'uso dell'istruzione `with`. Osservare il codice seguente in cui viene utilizzata l'istruzione `with`:

```
this.attachMovie("circleClip", "circle1Clip", 1);
with (circle1Clip) {
 _x = 20;
 _y = Math.round(Math.random()*20);
 _alpha = 15;
 createTextField("labelTxt", 100, 0, 20, 100, 22);
 labelTxt.text = "Circle 1";
 someVariable = true;
}
```

In questo codice viene associata un'istanza di un clip filmato dalla libreria e ne vengono modificate le proprietà tramite l'istruzione `with`. Se l'area di validità di una variabile non viene specificata, non è sempre chiaro in che punto le proprietà vengono impostate e pertanto il codice può risultare poco chiaro. Nel codice seguente, si potrebbe supporre che `someVariable` venga impostata all'interno del clip filmato `circle1Clip`, mentre in realtà viene impostata nella linea temporale del file SWF.

Se l'area di validità delle variabili viene impostata in modo esplicito, è più semplice comprendere le azioni eseguite dal codice e non occorre basarsi sull'istruzione `with`.

Nell'esempio seguente viene presentata una porzione di codice ActionScript leggermente più lunga, ma scritta in modo migliore, in cui vengono specificate le aree di validità delle variabili:

```
this.attachMovie("circleClip", "circle1Clip", 1);
circle1Clip._x = 20;
circle1Clip._y = Math.round(Math.random()*20);
circle1Clip._alpha = 15;
circle1Clip.createTextField("labelTxt", 100, 0, 20, 100, 22);
circle1Clip.labelTxt.text = "Circle 1";
circle1Clip.someVariable = true;
```

Un'eccezione alla regola si verifica se si lavora con l'API di disegno per disegnare forme, potrebbero essere necessarie diverse chiamate simili agli stessi metodi, ad esempio `lineTo` o `curveTo`, a causa delle funzionalità dell'API. Quando, ad esempio, si disegna un rettangolo semplice, sono necessarie quattro chiamate separate al metodo `lineTo`, come illustrato nel codice seguente:

```
this.createEmptyMovieClip("rectangleClip", 1);
with (rectangleClip) {
 lineStyle(2, 0x000000, 100);
 beginFill(0xFF0000, 100);
 moveTo(0, 0);
 lineTo(300, 0);
 lineTo(300, 200);
 lineTo(0, 200);
 lineTo(0, 0);
 endFill();
}
```

Se ogni metodo `lineTo` o `curveTo` fosse scritto con un nome di istanza completo, il codice risulterebbe disordinato e sarebbe difficile leggerlo ed eseguirne il debug.

## Informazioni sull'uso delle funzioni

Riutilizzare i blocchi di codice, se possibile, ad esempio chiamando una funzione più volte anziché creare porzioni diverse di codice ogni volta. Le funzioni possono essere rappresentate da porzioni di codice generiche, pertanto è possibile utilizzare gli stessi blocchi di codice per scopi leggermente diversi in un file SWF. Il riutilizzo del codice consente di creare applicazioni efficienti e ridurre la quantità di codice ActionScript 2.0 da scrivere e, di conseguenza, il tempo di sviluppo. È possibile creare funzioni su una linea temporale, in un file di classe o scrivere codice ActionScript in un componente basato su codice.

Se si utilizza ActionScript 2.0, non scrivere funzioni su una linea temporale; ma, se possibile, inserirle nei file di classe, come illustrato nell'esempio seguente:

```
class Circle {
 public function area(radius:Number):Number {
 return (Math.PI*Math.pow(radius, 2));
 }
 public function perimeter(radius:Number):Number {
 return (2 * Math.PI * radius);
 }
 public function diameter(radius:Number):Number {
 return (radius * 2);
 }
}
```

Utilizzare la sintassi seguente per la creazione di funzioni:

```
function myCircle(radius:Number):Number {
 //...
}
```

Non utilizzare la sintassi seguente, perché di difficile lettura:

```
myCircle = function(radius:Number):Number {
 //...
}
```

Nell'esempio seguente le funzioni vengono inserite in un file di classe. Si tratta di una procedura consigliata se si utilizza ActionScript 2.0, perché garantisce la riutilizzabilità del codice. Per riutilizzare le funzioni in altre applicazioni, è possibile importare la classe esistente anziché riscrivere il codice da zero oppure duplicare le funzioni nella nuova applicazione.

```
class mx.site.Utils {
 static function randomRange(min:Number, max:Number):Number {
 if (min>max) {
 var temp:Number = min;
 min = max;
 max = temp;
 }
 return (Math.floor(Math.random()*(max-min+1))+min);
 }
 static function arrayMin(numArr:Array):Number {
 if (numArr.length == 0) {
 return Number.NaN;
 }
 numArr.sort(Array.NUMERIC | Array.DESCENDING);
 var min:Number = Number(numArr.pop());
 return min;
 }
 static function arrayMax(numArr:Array):Number {
 if (numArr.length == 0) {
 return undefined;
 }
 numArr.sort(Array.NUMERIC);
 var max:Number = Number(numArr.pop());
 return max;
 }
}
```

Per utilizzare queste funzioni, aggiungere il codice ActionScript seguente al file FLA:

```
import mx.site.Utils;
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: "+randomMonth);
trace("min: "+min);
trace("max: "+max);
```

## Informazioni sull'interruzione di ripetizioni nel codice

Il gestore di eventi `onEnterFrame` è utile poiché può essere utilizzato da Flash per impostare ripetizioni di codice alla frequenza dei fotogrammi di un file SWF. Limitare tuttavia il più possibile il livello di ripetizioni utilizzate in un file Flash per non causare un degrado delle prestazioni. Se è presente, ad esempio, una porzione di codice che viene ripetuta ogni volta che l'indicatore di riproduzione raggiunge un fotogramma, l'operazione richiede molte risorse del processore e potrebbe generare problemi di prestazioni sui computer su cui viene riprodotto il file SWF. Se si fa ricorso al gestore di eventi `onEnterFrame` per ogni tipo di informazione o ripetizione nei file SWF, eliminare il gestore di eventi `onEnterFrame` quando non è più necessario. Nel codice ActionScript 2.0 seguente, si interrompe la ripetizione eliminando il gestore di eventi `onEnterFrame`:

```
circleClip.onEnterFrame = function() {
 circleClip._alpha -= 5;
 if (circleClip._alpha<=0) {
 circleClip.unloadMovie();
 delete this.onEnterFrame;
 trace("deleted onEnterFrame");
 }
};
```

In modo analogo, limitare il più possibile l'uso di `setInterval` e non dimenticare di eliminare l'intervallo quando non è più necessario per ridurre i requisiti di processore per il file SWF.

# Ottimizzazione di ActionScript e di Flash Player

Se si compila un file SWF contenente codice ActionScript 2.0 con Impostazioni pubblicazione impostato su Flash Player 6 e ActionScript 1.0, il codice funziona, a condizione che non utilizzi classi ActionScript 2.0. Il codice non prevede la distinzione tra maiuscole e minuscole, soltanto Flash Player. Tuttavia, se il file SWF viene compilato con Impostazioni pubblicazione impostato su Flash Player 7 o 8 e ActionScript 1.0, Flash applica la distinzione tra maiuscole e minuscole.

Le annotazioni di tipo (tipizzazione forte dei dati) vengono applicate al momento della compilazione per Flash Player 7 e 8 quando sono impostate le impostazioni di pubblicazione su ActionScript 2.0.

ActionScript 2.0 compile nel codice di byte di ActionScript 1.0 quando si pubblicano le applicazioni e quindi è possibile utilizzare Flash Player 6, 7 o 8 lavorando con ActionScript 2.0. Per ulteriori informazioni sull'ottimizzazione delle applicazioni, vedere “[Ottimizzazione del codice](#)”.

## Ottimizzazione del codice

Quando si ottimizza il codice, tenere presenti le linee guida seguenti:

- Non chiamare una funzione più volte in un ciclo.  
È preferibile includere il contenuto di una funzione di piccole dimensioni all'interno del ciclo.
- Fare ricorso a funzioni native, se possibile.  
L'esecuzione delle funzioni native è più rapida rispetto a quella delle funzioni definite dall'utente.
- Limitare il ricorso al tipo Object.  
Le annotazioni di tipo devono essere precise per migliorare le prestazioni. Utilizzare il tipo Object solo in mancanza di alternative.
- Non utilizzare la funzione `eval()` o l'operatore di accesso agli array.  
L'impostazione del riferimento locale è spesso preferibile e più efficiente.
- Assegnare la proprietà `Array.length` a una variabile prima di un ciclo.  
Assegnare la proprietà `Array.length` a una variabile prima che un ciclo la usi come condizione, invece di usare `myArr.length` stesso. Ad esempio,

```
var fontArr:Array = TextField.getFontList();
var arrayLen:Number = fontArr.length;
```

```
for (var i:Number = 0; i < arrayLen; i++) {
 trace(fontArr[i]);
}
```

invece di:

```
var fontArr:Array = TextField.getFontList();
for (var i:Number = 0; i < fontArr.length; i++) {
 trace(fontArr[i]);
}
```

- Ottimizzare soprattutto i cicli e qualsiasi azione ripetuta.  
L'elaborazione dei cicli con Flash Player richiede più tempo (come quelli che usano la funzione `setInterval()`).
- Aggiungere la parola chiave `var` nella dichiarazione di una variabile
- Non utilizzare variabili di classe o variabili globali quando sono sufficienti le variabili locali.

## Formattazione della sintassi ActionScript

La formattazione del codice ActionScript 2.0 secondo una procedura standardizzata è essenziale se si desidera scrivere codice comprensibile e gestibile, oltre che facile da comprendere e modificare da parte di altri sviluppatori. Sarebbe, ad esempio, molto difficile seguire la logica di un file FLA che non contiene commenti o rientri e per cui sono state adottate convenzioni per l'assegnazione dei nomi e la formattazione incoerenti. Se vengono applicati rientri ai blocchi di codice, ad esempio ai cicli e alle istruzioni `if`, il codice risulta più facile da leggere ed è più semplice eseguirne il debug.

Per ulteriori informazioni sulla formattazione del codice, consultare i seguenti argomenti:

- “Linee guida generali sulla formattazione” a pagina 826
- “Scrittura di istruzioni condizionali” a pagina 828
- “Scrittura di istruzioni composte” a pagina 830
- “Scrittura di un'istruzione `for`” a pagina 831
- “Scrittura di istruzioni `while` e `do..while`” a pagina 831
- “Scrittura di istruzioni `return`” a pagina 831
- “Scrittura di istruzioni `switch`” a pagina 832
- “Scrittura delle istruzioni `try..catch` e `try..catch..finally`” a pagina 833
- “Informazioni sull'uso della sintassi dei listener” a pagina 833

## Linee guida generali sulla formattazione

Quando si utilizzano spazi, interruzioni di riga e rientri con tabulazioni per aggiungere spazi nel codice, si migliora la leggibilità del codice stesso. L'uso della spaziatura migliora la leggibilità perché facilita l'illustrazione della gerarchia del codice. Rendere ActionScript 2.0 più facile da comprendere migliorando la leggibilità è un aspetto importante sia per gli studenti che per gli utenti avanzati che lavorano su progetti complessi. La leggibilità è critica anche durante il debug di ActionScript, perché se il codice è formattato correttamente e la spaziatura è idonea, è molto più semplice individuare gli errori.

Una porzione di codice ActionScript 2.0 può essere formattata o scritta in modi diversi. Esistono differenze nel modo in cui la sintassi viene formattata su più righe nell'Editor di ActionScript (il pannello Azioni o la finestra Script). È possibile, ad esempio, inserire parentesi graffe ({} ) o parentesi di altro tipo [( )] in posizioni diverse.

Per migliorare la leggibilità del codice ActionScript, Macromedia consiglia di adottare le procedure di formattazione illustrate di seguito.

- Inserire una riga vuota per separare i paragrafi (moduli) di ActionScript.

Per paragrafi di ActionScript si intendono sezioni di codice correlate dal punto di vista logico. L'aggiunta di una riga vuota tra i paragrafi, facilita agli utenti la lettura del codice ActionScript e una migliore comprensione della logica.

- Utilizzare i rientri in modo coerente per aiutare a evidenziare la gerarchia della struttura del codice.

Utilizzare gli stessi stili di rientro in tutto il codice ActionScript e allineare le parentesi ({} ) in modo corretto per migliorare la leggibilità del codice. Se la sintassi del codice ActionScript è corretta, Flash rientra automaticamente il codice in modo corretto quando l'utente preme Invio in Windows o A capo in Macintosh. Per inserire rientri nel codice ActionScript, se la sintassi è corretta, è possibile anche fare clic sul pulsante Formattazione automatica nell'Editor di ActionScript (il pannello Azioni o la finestra Script).

- Utilizzare interruzioni di riga per semplificare la lettura di istruzioni complesse.

Alcune istruzioni, ad esempio le istruzioni condizionali, possono essere formattate in modi diversi. A volte, se le istruzioni vengono formattate su più righe anziché su una riga sola, il codice risulta più facile da leggere.

- Includere uno spazio dopo una parola chiave seguita da parentesi [( )].

Nel codice ActionScript seguente ne viene illustrato un esempio:

```
do {
 // Un'azione
} while (condition);
```

- Non inserire uno spazio tra il nome di un metodo e una parentesi.

Nel codice ActionScript seguente ne viene illustrato un esempio:

```
function checkLogin():Boolean {
 // Istruzioni;
}
checkLogin();

o

printSize("size is " + foo + "\n");
```

- Inserire uno spazio dopo le virgolette in un elenco di argomenti.

L'utilizzo di spazi dopo le virgolette rende più semplice distinguere tra le chiamate dei metodi e le parole chiave, come illustrato nell'esempio seguente:

```
function addItems(item1:Number, item2:Number):Number {
 return (item1 + item2);
}
var sum:Number = addItems(1, 3);
```

- Usare spazi per separare tutti gli operatori e gli operandi.

L'utilizzo di spazi rende più semplice distinguere tra le chiamate dei metodi e le parole chiave, come illustrato nell'esempio seguente:

```
// Corretto
var sum:Number = 7 + 3;
// Scorretto
var sum:Number=7+3;
```

Un'eccezione alla regola è rappresentata dall'operatore punto (.).

- Non inserire uno spazio tra gli operatori unari e i relativi operandi.

Ad esempio, incremento (++) e decremento (--), come illustrato nell'esempio seguente:

```
while (d++ = s++)
-2, -1, 0
```

- Non inserire spazi dopo una parentesi aperta e prima di una parentesi chiusa.

Nel codice ActionScript seguente ne viene illustrato un esempio:

```
// Scorretto
("size is " + foo + "\n");
// Corretto
("size is " + foo + "\n");
```

- Per migliorare la leggibilità del codice ActionScript inserire una sola istruzione per riga

Nel codice ActionScript seguente ne viene illustrato un esempio:

```
theNum++; // Corretto
theOtherNum++; // Corretto
aNum++; anotherNum++; // Scorretto
```

- Non incorporare le assegnazioni.

Questa procedura viene a volte adottata per migliorare le prestazioni di un file SWF in fase di runtime, perché è più difficile da leggere ed eseguirne il debug. Il codice ActionScript seguente ne illustra un esempio (evitare tuttavia di utilizzare nomi a carattere singolo nel codice effettivo):

```
var myNum:Number = (a = b + c) + d;
```

- Assegnare le variabili in istruzioni separate.

Il codice ActionScript seguente ne illustra un esempio (evitare tuttavia di utilizzare nomi a carattere singolo nel codice effettivo):

```
var a:Number = b + c;
var myNum:Number = a + d;
```

- Interrompere una riga prima di un operatore.
- Interrompere una riga dopo una virgola.
- Allineare la seconda riga con l'inizio dell'espressione presente sulla riga di codice successiva.



Per controllare le impostazioni di rientro automatico e rientro, selezionare Modifica > Preferenze (Windows) o Flash > Preferenze (Macintosh) e quindi selezionare la scheda ActionScript.

## Scrittura di istruzioni condizionali

Adottare le linee guida seguenti per la scrittura di istruzioni condizionali:

- Inserire le condizioni su righe separate nelle istruzioni if, else..if e if..else.
- Utilizzare parentesi graffe ({} ) per le istruzioni if.
- Formattare le parentesi graffe come illustrato negli esempi seguenti:

```
// Istruzione if
if (condition) {
 // istruzioni
}

// istruzione if..else
if (condition) {
 // istruzioni
} else {
 // istruzioni
}

// istruzione else..if
if (condition) {
 // istruzioni
```

```
 } else if (condition) {
 // istruzioni
 } else {
 // istruzioni
 }
```

Quando si scrivono condizioni complesse, utilizzare le parentesi tonde [ ] per raggruppare le condizioni. Se non si utilizzano le parentesi, si potrebbero verificare errori di priorità degli operatori.

Nel codice seguente, ad esempio, non vengono utilizzate le parentesi per le condizioni:

```
if (fruit == apple && veggie == leek) {}
```

Nel codice seguente la sintassi è corretta in quanto vengono aggiunte le parentesi per le condizioni:

```
if ((fruit == apple) && (veggie == leek)) {}
```

È possibile scrivere un'istruzione condizionale che restituisca un valore booleano in due modi.

Il secondo esempio è preferibile:

```
if (cartArr.length>0) {
 return true;
} else {
 return false;
}
```

Confrontare questo esempio con il precedente:

```
// Migliore
return (cartArr.length > 0);
```

Il secondo frammento di codice è più breve e il numero di espressioni da valutare è inferiore; è quindi più semplice da leggere e comprendere.

L'esempio seguente verifica se la variabile *y* è maggiore di zero (0) e restituisce il risultato di *x/y* o il valore 0:

```
return ((y > 0) ? x/y : 0);
```

Nell'esempio di codice seguente viene eseguita la stessa operazione, ma è scritto in modo diverso. Questo esempio è preferibile:

```
if (y>0) {
 return x/y;
} else {
 return 0;
}
```

La sintassi abbreviata dell'istruzione *if* del primo esempio è detta operatore condizionale (?:) e consente di convertire istruzioni *if..else* semplici in una sola riga di codice. In questo caso, una sintassi più breve riduce la leggibilità.

Se si devono utilizzare operatori condizionali, inserire la condizione iniziale (prima del punto di domanda [?]) all'interno di parentesi per migliorare la leggibilità del codice. Il frammento di codice precedente ne è un esempio.

## Scrittura di istruzioni composte

Le istruzioni composte contengono un elenco di istruzioni all'interno di parentesi graffe ({}). Le istruzioni all'interno di queste parentesi sono rientrate rispetto all'istruzione composta. Nel codice ActionScript seguente ne viene illustrato un esempio:

```
if (a == b) {
 // Questo codice è rientrato
 trace("a == b");
}
```

Inserire le parentesi graffe prima e dopo ogni istruzione quando l'istruzione fa parte di una struttura di controllo, ovvero if..else o for, anche se la struttura contiene una sola istruzione. Di seguito è riportato un esempio di codice scritto in modo poco chiaro:

```
// Scorretto
if (numUsers == 0)
 trace("no users found.");
```

Sebbene il codice non generi errori, il formato non è considerato corretto perché non sono presenti le parentesi graffe prima e dopo le istruzioni. In questo caso, se viene aggiunta un'ulteriore istruzione dopo l'istruzione trace, questa verrà eseguita indipendentemente dal fatto che la variabile numUsers sia uguale a 0:

```
// Scorretto
var numUsers:Number = 5;
if (numUsers == 0)
 trace("no users found.");
 trace("I will execute");
```

L'esecuzione del codice nonostante la variabile numUsers può portare a risultati imprevisti. Per questo motivo, aggiungere le parentesi graffe come illustrato nell'esempio seguente:

```
var numUsers:Number = 0;
if (numUsers == 0) {
 trace("no users found");
}
```

Quando si scrive una condizione, non aggiungere l'istruzione ==true ridondante al codice, come segue:

```
if (something == true) {
 // istruzioni
}
```

Se viene effettuato un confronto con `false`, è possibile utilizzare `if (something==false)` o `if(!something)`.

## Scrittura di un'istruzione for

È possibile scrivere l'istruzione `for` nel formato seguente:

```
for (init; condition; update) {
 // istruzioni
}
```

La struttura seguente illustra l'istruzione `for`:

```
var i:Number;
for (var i = 0; i<4; i++) {
 myClip.duplicateMovieClip("newClip" + i + "Clip", i + 10, {_x:i*100,
 _y:0});
}
```

Inserire uno spazio dopo ogni espressione all'interno di un'istruzione `for`.

## Scrittura di istruzioni while e do..while

È possibile scrivere le istruzioni `while` nel formato seguente:

```
while (condition) {
 // istruzioni
}
```

È possibile scrivere le istruzioni `do..while` nel formato seguente:

```
do {
 // istruzioni
} while (condition);
```

## Scrittura di istruzioni return

Non utilizzare le parentesi tonde `[]()` con le istruzioni `return` che dispongono di valori.

Utilizzare le parentesi con le istruzioni `return` solo se serve per rendere il valore più chiaro, come illustrato nella terza riga del frammento di codice ActionScript seguente:

```
return;
return myCar.paintColor;
// Parentesi utilizzate per rendere più chiaro il valore restituito
return ((paintColor)? paintColor: defaultColor);
```

## Scrittura di istruzioni switch

- Tutte le istruzioni `switch` comprendono un'etichetta `case` predefinita. Il `case` predefinita è l'ultima in un'istruzione `switch`. Il `case default` comprende a sua volta un'istruzione `break` che evita di uscire dall'istruzione `switch` quando viene soddisfatta la condizione.
- Se un blocco `case` non contiene un'istruzione `break`, si verifica una condizione che viene detta *fall through* (vedere `case A` nell'esempio seguente). L'istruzione deve contenere un commento al posto dell'istruzione `break`, come si può notare nell'esempio seguente dopo `case A`. In questo esempio, se la condizione restituisce `case A`, vengono eseguite le istruzioni `case A` e `case B`.

È possibile scrivere le istruzioni `switch` nel formato seguente:

```
switch (condition) {
 case A :
 // istruzioni
 // Continua la valutazione
 case B :
 // istruzioni
 break;
 case Z :
 // istruzioni
 break;
 default :
 // istruzioni
 break;
}
```

## Scrittura delle istruzioni try..catch e try..catch..finally

Scrivere le istruzioni try..catch e try..catch..finally con il formato seguente:

```
var myErr:Error;
// try..catch
try {
 // istruzioni
} catch (myErr) {
 // istruzioni
}

try..catch..finally
try {
 // istruzioni
} catch (myErr) {
 // istruzioni
} finally {
 // istruzioni
}
```

## Informazioni sull'uso della sintassi dei listener

I listener di eventi in Flash 8 possono essere scritti in modi diversi. Negli esempi di codice seguenti vengono adottate le tecniche più comuni. Nel primo esempio è presente la sintassi di un listener con formattazione corretta in cui viene utilizzato un componente Loader per caricare contenuto in un file SWF. L'evento progress viene generato quando il contenuto viene caricato e l'evento complete indica il termine del caricamento.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();
ldrListener.progress = function(evt:Object) {
 trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
};
ldrListener.complete = function(evt:Object) {
 trace("loader complete:" + evt.target._name);
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Per variare leggermente il primo esempio presentato in questa sezione è possibile utilizzare il metodo `handleEvent`, sebbene questa tecnica risulti leggermente meno pratica. Questa tecnica non è consigliata in quanto richiede l'uso di una serie di istruzioni `if..else` o di un'istruzione `switch` per individuare l'evento rilevato.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();

ldrListener.handleEvent = function(evt:Object) {
 switch (evt.type) {
 case "progress" :
 trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
 break;
 case "complete" :
 trace("loader complete:" + evt.target._name);
 break;
 }
};

boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

# Messaggi di errore

A

Macromedia Flash Basic 8 e Macromedia Flash Professional 8 garantiscono la segnalazione degli errori in fase di compilazione se i file vengono pubblicati per ActionScript 2.0 (impostazione predefinita). La tabella seguente contiene l'elenco dei messaggi di errore che possono essere generati dal compilatore di Flash:

<b>Numero errore</b>	<b>Testo del messaggio</b>
1093	Previsto nome di classe.
1094	È previsto un nome di classe base dopo la parola chiave 'extends'.
1095	Un attributo membro non è stato utilizzato correttamente.
1096	Impossibile ripetere più di una volta lo stesso nome del membro.
1097	A tutte le funzioni membro deve essere stato assegnato un nome.
1099	Istruzione non consentita in una definizione di classe.
1100	Classe o interfaccia già definita con questo nome.
1101	Tipo non corrispondente.
1102	Nessuna classe denominata '<ClassName>'.
1103	Nessuna proprietà denominata '<propertyName>'.
1104	Si è tentato di effettuare una chiamata di funzione su una non funzione.
1105	Tipo non corrispondente nell'istruzione di assegnazione: è stato trovato [tipo-lhs] mentre è richiesto [tipo-rhs].
1106	Il membro è privato. Impossibile accedervi.
1107	Le dichiarazioni di variabili non sono consentite nelle interfacce.
1108	Le dichiarazioni di eventi non sono consentite nelle interfacce.
1109	Le dichiarazioni getter/setter non sono consentite nelle interfacce.
1110	I membri privati non sono consentiti nelle interfacce.
1111	I corpi delle funzioni non sono consentiti nelle interfacce.

---

<b>Numero errore</b>	<b>Testo del messaggio</b>
1112	Una classe non può estendere se stessa.
1113	Un'interfaccia non può estendere se stessa.
1114	Nessuna interfaccia definita con questo nome.
1115	Una classe non può estendere un'interfaccia.
1116	Un'interfaccia non può estendere una classe.
1117	È previsto un nome di interfaccia dopo la parola chiave 'implements'.
1118	Una classe non può implementare una classe, ma solo interfacce.
1119	La classe deve implementare il metodo 'methodName' dall'interfaccia 'interfaceName'.
1120	L'implementazione di un metodo di interfaccia deve essere un metodo, non una proprietà.
1121	Una classe non può estendere la stessa interfaccia più di una volta.
1122	L'implementazione del metodo di interfaccia non corrisponde alla relativa definizione.
1123	Il costrutto è disponibile solo in ActionScript 1.0.
1124	Il costrutto è disponibile solo in ActionScript 2.0.
1125	Membri statici non consentiti nelle interfacce.
1126	L'espressione restituita deve corrispondere al tipo restituito della funzione.
1127	Istruzione di restituzione necessaria in questa funzione.
1128	Attributo utilizzato al di fuori della classe.
1129	Una funzione con tipo restituito Void non può restituire un valore.
1130	La proposizione 'extends' deve apparire prima della proposizione 'implements'.
1131	È previsto un identificatore del tipo dopo ':'.
1132	Le interfacce devono utilizzare la parola chiave 'extends', non 'implements'.
1133	Una classe non può estendere più di una classe.
1134	Un'interfaccia non può estendere più di un'interfaccia.
1135	Nessun metodo denominato '<methodName>'.
1136	Istruzione non consentita in una definizione delle interfacce.
1137	Una funzione set richiede esattamente un parametro.
1138	Una funzione get non richiede parametri.

---

---

<b>Numero errore</b>	<b>Testo del messaggio</b>
1139	È possibile definire le classi solo in script di classi ActionScript 2.0 esterni.
1140	Gli script di classi ActionScript 2.0 possono definire solo costrutti di classi o interfacce.
1141	Il nome della classe, '<A.B.C>', è in conflitto con il nome di un'altra classe caricata, '<A.B>'. Questo errore si verifica quando il compilatore di ActionScript 2.0 non è in grado di compilare una classe perché il nome completo di una classe esistente corrisponde in parte al nome della classe che provoca il conflitto. La compilazione della classe <code>mx.com.util</code> , ad esempio, genera l'errore 1141 se la classe <code>mx.com</code> è una classe compilata.
1142	Impossibile caricare la classe o l'interfaccia '<Class or Interface Name>'.
1143	È possibile definire le interfacce solo in script di classi ActionScript 2.0 esterni.
1144	Impossibile accedere alle variabili di istanza nelle funzioni statiche.
1145	Impossibile annidare le definizioni di classe e interfaccia.
1146	La proprietà a cui si fa riferimento non ha l'attributo statico.
1147	La chiamata a super non corrisponde al supercostruttore.
1148	Solo l'attributo pubblico è consentito per i metodi di interfaccia.
1149	Impossibile utilizzare la parola chiave import come direttiva.
1150	Esportare il filmato Flash come Flash 7 per usare questa azione.
1151	Esportare il filmato Flash come Flash 7 per usare questa espressione.
1152	La proposizione di eccezione non è collocata correttamente.
1153	Una classe deve avere solo un costruttore.
1154	Un costruttore non può restituire un valore.
1155	Un costruttore non può specificare un tipo restituito.
1156	Una variabile non può essere del tipo Void.
1157	Un parametro della funzione non può essere del tipo Void.
1158	È possibile accedere direttamente ai membri statici solo attraverso le classi.
1159	Le interfacce multiple implementate contengono lo stesso metodo con tipi differenti.
1160	Una classe o un'interfaccia definita con questo nome esiste già.
1161	Impossibile eliminare le classi, le interfacce e i tipi incorporati.

---

---

<b>Numero errore</b>	<b>Testo del messaggio</b>
1162	Nessuna classe con questo nome.
1163	La parola chiave '<keyword>' è riservata per ActionScript 2.0 e non può essere utilizzata in questo contesto.
1164	Definizione degli attributi personalizzata non terminata.
1165	È possibile definire solo una classe o interfaccia per file .as di ActionScript 2.0.
1166	La classe in fase di compilazione, '<A.b>', non corrisponde alla classe importata, '<A.B>'. Questo errore si verifica quando il nome di una classe viene scritto con maiuscole o minuscole diverse rispetto a una classe importata. La compilazione della classe <code>mx.com.util</code> , ad esempio, genera l'errore 1166 se nel file <code>util.as</code> è presente la classe <code>import mx.Com</code> .
1167	Immettere un nome di classe.
1168	Il nome di classe immesso contiene un errore di sintassi.
1169	Il nome di interfaccia immesso contiene un errore di sintassi.
1170	Il nome di classe base immesso contiene un errore di sintassi.
1171	Il nome di interfaccia base immesso contiene un errore di sintassi.
1172	Immettere un nome di interfaccia.
1173	Immettere un nome di classe o interfaccia.
1174	Il nome di classe o interfaccia immesso contiene un errore di sintassi.
1175	'variable' non accessibile da questa area.
1176	Sono state trovate occorrenze multiple dell'attributo 'get/set/private/public/static'.
1177	Un attributo della classe non è stato utilizzato correttamente.
1178	Impossibile utilizzare le funzioni e le variabili di istanza per inizializzare le variabili statiche.
1179	Circularità runtime individuate tra le seguenti classi: <elenco delle classi definite dall'utente>. Questo errore di runtime indica che le classi personalizzate fanno riferimento ad altre classi personalizzate in modo scorretto.
1180	Il lettore Flash Player di destinazione non supporta il debug.
1181	Il lettore Flash Player di destinazione non supporta l'evento <code>releaseOutside</code> .
1182	Il lettore Flash Player di destinazione non supporta l'evento <code>dragOver</code> .

---

---

<b>Numero errore</b>	<b>Testo del messaggio</b>
1183	Il lettore Flash Player di destinazione non supporta l'evento dragOut.
1184	Il lettore Flash Player di destinazione non supporta le azioni di trascinamento.
1185	Il lettore Flash Player di destinazione non supporta l'azione loadMovie.
1186	Il lettore Flash Player di destinazione non supporta l'azione getUrl.
1187	Il lettore Flash Player di destinazione non supporta l'azione FSCommand.
1188	Istruzioni di importazione non consentite all'interno delle definizioni di classe o interfaccia.
1189	Impossibile importare la classe '<A.B>'. Il nome foglia è già in fase di risoluzione per la classe definita, '<C.B>'. La compilazione della classe <code>util</code> , ad esempio, genera l'errore 1189 se nel file <code>util.as</code> è presente l'istruzione <code>import mx.util</code> .
1190	Impossibile importare la classe '<A.B>'. Il nome foglia è già in fase di risoluzione per la classe definita, '<C.B>'. La compilazione della classe <code>import jv.util</code> , ad esempio, genera l'errore 1190 se nel file AS è presente anche l'istruzione <code>import mx.util</code> .
1191	Le variabili di istanza di una classe possono essere inizializzate solo per espressioni costanti di tipo compile time.
1192	Le funzioni dei membri di una classe non possono avere lo stesso nome di una funzione di costruttore di una superclasse.
1193	Il nome della classe, '<ClassName>', è in conflitto con il nome di un'altra classe caricata.
1194	È necessario chiamare il supercostruttore per primo nel corpo del costruttore.
1195	L'identificatore '<className>' non risolverà l'oggetto incorporato '<ClassName>' durante il runtime.
1196	È necessario definire la classe '<A.B.ClassName>' in un file il cui percorso relativo è '<A.B>'.
1197	Il carattere jolly '*' non è utilizzato correttamente nel nome di classe '<ClassName>'.
1198	La funzione membro '<classname>' è diversa per maiuscole/minuscole rispetto al nome della classe definita, '<ClassName>', e non viene trattata come costruttore della classe in fase di runtime.
1199	L'unico tipo consentito per un iteratore di un ciclo for-in è String.
1200	Una funzione setter non può restituire un valore.

---

---

<b>Numero errore</b>	<b>Testo del messaggio</b>
1201	Gli unici attributi consentiti per le funzioni di costruzione sono pubblico e privato.
1202	Impossibile trovare il file 'toplevel.as' richiesto per il controllo di assegnazione di ActionScript 2.0. Assicurarsi che la directory '\$(LocalData)/Classes' sia elencata all'interno del percorso di classe globale delle preferenze di ActionScript.
1203	Il ramo tra <spanStart> e <spanEnd> supera i 32 K.
1204	Nessuna classe o pacchetto con il nome '<packageName>' trovata in '<PackageName>'.
1205	Il lettore Flash Player di destinazione non supporta l'azione FSCommand2.
1206	La funzione del membro '<functionName>' supera i 32 K.
1207	La funzione anonima intorno alla linea <lineNumber> supera i 32 K.
1208	Il codice intorno alla linea <lineNumber> supera i 32 K.
1210	Il nome di pacchetto '<PackageName>' non può essere utilizzato anche come nome di metodo.
1211	Il nome di pacchetto '<PackageName>' non può essere utilizzato anche come nome di proprietà.
1212	Impossibile creare il file ASO per la classe '<ClassName>'. Verificare che il nome completo della classe sia abbastanza corto, in modo che il nome del file ASO, '<ClassName.aso>', sia inferiore a 255 caratteri.
1213	Questo tipo di virgoletta non è consentito in ActionScript. Cambiarlo con una virgoletta doppia (diritta) standard.

---

# Operatori di Flash 4 obsoleti

B

Nella tabella seguente sono elencati gli operatori di Flash 4 dichiarati obsoleti in ActionScript 2.0. Utilizzare questi operatori solo se si esegue la pubblicazione per Flash Player 4 e versioni precedenti.

<b>Operatore</b>	<b>Descrizione</b>	<b>Associatività</b>
not	NOT logico	Da destra a sinistra
and	AND logico	Da sinistra a destra
or	OR logico (stile Flash 4)	Da sinistra a destra
add	Concatenazione di stringhe (in precedenza &)	Da sinistra a destra
instanceof	Istanza di	Da sinistra a destra
lt	Minore o uguale a (versione per stringhe)	Da sinistra a destra
le	Minore o uguale a (versione per stringhe)	Da sinistra a destra
gt	Maggiore o uguale a (versione per stringhe)	Da sinistra a destra
ge	Maggiore o uguale a (versione per stringhe)	Da sinistra a destra
eq	Uguale (versione per stringhe)	Da sinistra a destra
ne	Non uguale (versione per stringhe)	Da sinistra a destra



# Tasti della tastiera e valori dei codici tasto

C

Le tabelle seguenti contengono l'elenco di tutti i tasti di una tastiera standard e i valori di codice tasto corrispondenti, nonché i valori ASCII, utilizzati per identificare i tasti in ActionScript:

- “Lettere dalla A alla Z e numeri standard da 0 a 9” a pagina 844
- “Tasti del tastierino numerico” a pagina 846
- “Tasti funzione” a pagina 847
- “Altri tasti” a pagina 848

Per intercettare il comportamento incorporato relativo alla pressione dei tasti è possibile utilizzare le costanti della classe Key. Per ulteriori informazioni su on() handler, vedere on handler nella *Guida di riferimento di ActionScript 2.0*. Per catturare i valori dei codici tasto e i valori dei codici tasto ASCII mediante un file SWF e la pressione dei tasti, utilizzare il codice ActionScript seguente.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
 trace("GIÙ -> Codice: " + Key.getCode() + "\tASCII: " + Key.getAscii() +
 "\tKey: " + chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Per ulteriori informazioni sulla classe Key, vedere Key nella *Guida di riferimento di ActionScript 2.0*. Per registrare tasti durante la prova di un file SWF nell'ambiente di creazione (Controllo > Prova filmato), assicurarsi di selezionare Controllo > Disattiva tasti di scelta rapida.

## Lettere dalla A alla Z e numeri standard da 0 a 9

La tabella seguente elenca i tasti di una tastiera standard per le lettere dalla A alla Z e i numeri da 0 a 9 con i corrispondenti valori di codice tasto utilizzati per identificare i tasti in ActionScript:

Tasto della lettera o del numero	Codice tasto	Codice tasto ASCII
A	65	65
B	66	66
C	67	67
D	68	68
E	69	69
F	70	70
G	71	71
H	72	72
I	73	73
J	74	74
K	75	75
L	76	76
M	77	77
N	78	78
O	79	79
P	80	80
Q	81	81
R	82	82
S	83	83
T	84	84
U	85	85
V	86	86
W	87	87
X	88	88
Y	89	89

Tasto della lettera o del numero	Codice tasto	Codice tasto ASCII
Z	90	90
0	48	48
1	49	49
2	50	50
3	51	51
4	52	52
5	53	53
6	54	54
7	55	55
8	56	56
9	57	57
a	65	97
b	66	98
c	67	99
d	68	100
e	69	101
f	70	102
g	71	103
h	72	104
i	73	105
j	74	106
k	75	107
l	76	108
m	77	109
n	78	110
o	79	111
p	80	112
q	81	113
r	82	114

<b>Tasto della lettera o del numero</b>	<b>Codice tasto</b>	<b>Codice tasto ASCII</b>
s	83	115
t	84	116
u	85	117
v	86	118
w	87	119
x	88	120
y	89	121
z	90	122

## Tasti del tastierino numerico

La tabella seguente elenca i tasti del tastierino numerico con i valori corrispondenti dei codici tasto utilizzati per identificare i tasti in ActionScript:

<b>Tasto del tastierino numerico</b>	<b>Codice tasto</b>	<b>Codice tasto ASCII</b>
Tast. num 0	96	48
Tast. num 1	97	49
Tast. num 2	98	50
Tast. num 3	99	51
Tast. num 4	100	52
Tast. num 5	101	53
Tast. num 6	102	54
Tast. num 7	103	55
Tast. num 8	104	56
Tast. num 9	105	57
Moltiplicazione	106	42
Addizione	107	43
Invio	13	13
Sottrazione	109	45

<b>Tasto del tastierino numerico</b>	<b>Codice tasto</b>	<b>Codice tasto ASCII</b>
Separatore decimale	110	46
Divisione	111	47

## Tasti funzione

La tabella seguente contiene l'elenco dei tasti funzione di una tastiera standard con i valori corrispondenti dei codici tasto utilizzati per identificare i tasti in ActionScript:

<b>Tasto funzione</b>	<b>Codice tasto</b>	<b>Codice tasto ASCII</b>
F1	112	0
F2	113	0
F3	114	0
F4	115	0
F5	116	0
F6	117	0
F7	118	0
F8	119	0
F9	120	0
F10	Questo tasto è riservato dal sistema e non può essere utilizzato in ActionScript.	Questo tasto è riservato dal sistema e non può essere utilizzato in ActionScript.
F11	122	0
F12	123	0
F13	124	0
F14	125	0
F15	126	0

## Altri tasti

La tabella seguente contiene l'elenco dei tasti di una tastiera standard diversi dalle lettere, dai numeri, dai tasti del tastierino numerico e dai tasti funzione con i valori corrispondenti dei codici tasto utilizzati per identificare i tasti in ActionScript:

Tasto	Codice tasto	Codice tasto ASCII
Backspace	8	8
Tab	9	9
Enter	13	13
Maiusc	16	0
Ctrl	17	0
Bloc Maiusc	20	0
Esc	27	27
Barra spaziatrice	32	32
Pg su	33	0
Pg giù	34	0
Fine	35	0
Home	36	0
Freccia sinistra	37	0
Freccia su	38	0
Freccia destra	39	0
Freccia giù	40	0
Ins	45	0
Canc	46	127
Bloc Num	144	0
BloccScorr	145	0
Pausa/Interr	19	0
; :	186	59
= +	187	61
- _	189	45
/ ?	191	47
' ~	192	96

Tasto	Codice tasto	Codice tasto ASCII
[ {	219	91
\ \	220	92
] }	221	93
" "	222	39
,	188	44
.	190	46
/	191	47

Per ulteriori valori ASCII e di codici tasto, utilizzare ActionScript all'inizio di questa appendice e premere il tasto desiderato per trovare il relativo codice.



# Creazione di script per le versioni precedenti di Flash Player

ActionScript è stato notevolmente modificato con ogni versione degli strumenti di creazione di Macromedia Flash e Flash Player. Quando si crea contenuto per Macromedia flash Player 8, è possibile sfruttare a pieno tutte le funzionalità di ActionScript. È comunque possibile utilizzare Flash 8 per creare contenuto per le versioni precedenti di Flash Player. In questo caso, però, non è possibile avvalersi di tutti gli elementi del linguaggio.

Questo capitolo fornisce le linee guida per facilitare la creazione di script sintatticamente corretti per la versione di Flash Player che si desidera impostare come destinazione.



Per informazioni sulla versione di Flash Player più utilizzata nei diversi Paesi, vedere [www.macromedia.com/software/player\\_census/flashplayer/](http://www.macromedia.com/software/player_census/flashplayer/).

## Informazioni sulla sicurezza nelle versioni precedenti di Flash Player

Durante la creazione degli script, si consiglia di utilizzare le informazioni sulla disponibilità di ogni elemento presenti nella *Guida di riferimento di ActionScript* per determinare se l'elemento che si desidera utilizzare è supportato nella versione di Flash Player per cui si esegue la pubblicazione. È possibile inoltre stabilire quali elementi possono essere utilizzati visualizzando la casella degli strumenti Azioni; gli elementi non supportati nella versione impostata come destinazione sono evidenziati in giallo.

Se si crea contenuto per Flash Player 6 o 7, è consigliabile utilizzare ActionScript 2.0, in quanto garantisce numerose importanti funzionalità assenti in ActionScript 1.0, ad esempio una gestione più efficace degli errori del compilatore e una programmazione orientata agli oggetti più potente.

Per specificare la versione del lettore e di ActionScript da utilizzare per la pubblicazione di un documento, scegliere File > Impostazioni pubblicazione, quindi selezionare le opzioni desiderate nella scheda Flash. Per eseguire la pubblicazione per Flash Player 4, consultare la sezione successiva.

# Uso di Flash 8 per creare contenuto per Flash Player 4

Per utilizzare Flash 8 per creare contenuto per Flash Player 4, specificare Flash Player 4 nella scheda Flash della finestra di dialogo Impostazioni pubblicazione (File > Impostazioni pubblicazione).

Il linguaggio ActionScript di Flash Player 4 prevede un solo tipo di dati di base, che viene utilizzato sia per la gestione dei dati numerici, sia per la gestione dei dati di tipo stringa. Quando si crea un'applicazione per Flash Player 4, è necessario utilizzare gli operatori stringa obsoleti della categoria Obsoleto > Operatori nella casella degli strumenti ActionScript.

È possibile utilizzare inoltre le seguenti funzionalità di Flash 8 quando si pubblica per Flash Player 4:

- Operatore di accesso array e oggetti ([ ])
- Operatore punto (.)
- Operatori logici, operatori di assegnazione, operatori di incremento e decremento prima e dopo l'operazione
- Operatore modulo (%) e tutti i metodi e le proprietà della classe Math

Gli elementi del linguaggio indicati di seguito non sono supportati in Flash Player 4 all'origine. In Flash 8 vengono esportati come approssimazioni seriali che determinano la restituzione di risultati meno accurati dal punto di vista numerico. Inoltre, dato l'uso di approssimazioni seriali nel file SWF, questi elementi del linguaggio occupano più spazio nei file SWF di Flash Player 4 rispetto ai file SWF di Flash Player 5 e delle versioni successive.

- Azioni for, while, do..while, break e continue
- Azioni print() e printAsBitmap()
- Azione switch

Per ulteriori informazioni, vedere “[Informazioni sulla sicurezza nelle versioni precedenti di Flash Player](#)” a pagina 851.

## Uso di Flash 8 per aprire file Flash 4

Il linguaggio ActionScript di Flash 4 disponeva di un solo tipo di dati true: stringa. Utilizzava diversi tipi di operatori nelle espressioni per indicare se il valore doveva essere considerato come stringa o come numero. Nelle versioni successive di Flash, è possibile utilizzare un unico insieme di operatori su tutti i tipi di dati.

Quando in Flash 5 o una versione successiva si apre un file creato in Flash 4, le espressioni di ActionScript vengono convertite automaticamente per essere rese compatibili con la nuova sintassi. In Flash vengono eseguite le seguenti conversioni di tipi di dati e operatori:

- L'operatore = veniva usato in Flash 4 per esprimere l'uguaglianza numerica. In Flash 5 e nelle versioni successive, == è l'operatore di uguaglianza e = è l'operatore di assegnazione. Gli operatori = dei file Flash 4 vengono convertiti automaticamente in ==.
- In Flash, la conversione dei tipi di dati viene eseguita automaticamente per garantire il corretto funzionamento degli operatori. A causa dell'introduzione di tipi di dati multipli, gli operatori seguenti hanno assunto un nuovo significato:

+, ==, !=, <>, <, >, >=, <=

Nel linguaggio ActionScript di Flash 4 tali operatori erano sempre operatori numerici. In Flash 5 e nelle versioni successive, si comportano in modo differente a seconda dei tipi di dati degli operandi. Per evitare differenze semantiche nei file importati, viene applicata la funzione Number() a tutti gli operandi di tali operatori. (Ai numeri costanti non viene applicata la funzione Number(), in quanto sono ovviamente dei valori numerici). Per ulteriori informazioni su questi operatori, vedere la tabella degli operatori in

[“Informazioni sulla priorità e l'associatività degli operatori” a pagina 146](#) e [“Operatori di Flash 4 obsoleti” a pagina 841](#).

- In Flash 4, la sequenza di escape \n generava un carattere di ritorno a capo (ASCII 13). Per compatibilità con lo standard ECMA-262, in Flash 5 e nelle versioni successive \n genera un carattere di avanzamento riga (ASCII 10). Una sequenza di escape \n nei file FLA di Flash 4 viene automaticamente convertita in \r.
- L'operatore & veniva usato in Flash 4 per l'addizione di stringhe. In Flash 5 e nelle versioni successive, & è l'operatore AND bit a bit. Il nuovo operatore di addizione stringhe è denominato add. Tutti gli operatori & dei file di Flash 4 vengono convertiti automaticamente in operatori add.
- Molte funzioni di Flash 4, ad esempio Get Timer, Set Variable, Stop e Play, non richiedevano l'uso delle parentesi. Per garantire la coerenza della sintassi, la funzione getTimer e tutte le azioni richiedono ora la presenza di parentesi [()] che vengono aggiunte automaticamente durante la conversione.

- In Flash 5 e nelle versioni successive, se viene eseguita la funzione `getProperty` su un clip filmato che non esiste, non viene restituito 0, ma `undefined`. L'istruzione ActionScript `undefined == 0` è pertanto `false` nelle versioni successive a Flash 4. In Flash 4 si aveva invece `undefined == 1`. In caso di conversione di file di Flash 4, questo problema può essere risolto in Flash 5 e nelle versioni successive inserendo funzioni `Number()` nei confronti di uguaglianza. Nell'esempio seguente, `Number()` impone la conversione di `undefined` in 0 per consentire la corretta esecuzione del confronto:

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

**NOTA**

Se sono state utilizzate parole chiave di Flash 5 o di una versione successiva come nomi di variabili in ActionScript di Flash 4, viene restituito un errore di sintassi durante la compilazione in Flash 8. Per risolvere il problema, assegnare un nuovo nome alle variabili in tutte le posizioni. Per informazioni, vedere ["Informazioni sulle parole riservate"](#) a pagina 104 e ["Informazioni sull'assegnazione di nomi alle variabili"](#) a pagina 356.

## Uso della barra rovesciata

In Flash 3 e 4 la sintassi della barra (/) indicava il percorso target di un clip filmato o di una variabile. Secondo questa sintassi, al posto dei punti vengono utilizzate le barre e le variabili vengono precedute dai due punti, come nell'esempio seguente:

```
myMovieClip/childMovieClip:myVariable
```

Per scrivere lo stesso percorso target con la sintassi del punto supportata in Flash Player 5 e nelle versioni successive, è necessario utilizzare il codice seguente:

```
myMovieClip.childMovieClip.myVariable
```

La sintassi della barra veniva generalmente associata all'azione `tellTarget`, il cui uso è ora sconsigliato. Si preferisce infatti l'azione `with` per la sua maggiore compatibilità con la sintassi del punto. Per ulteriori informazioni, vedere `tellTarget` function e `with` statement nella *Guida di riferimento di ActionScript*.

# Programmazione orientata agli oggetti con ActionScript 1.0

Le informazioni presenti in questa appendice, tratte dalla documentazione di Macromedia Flash MX, forniscono indicazioni sull'uso del modello a oggetti di ActionScript 1.0 per la creazione di script. L'appendice viene fornita per i motivi seguenti:

- È necessario utilizzare ActionScript 1.0 se si desidera creare script orientati agli oggetti che supportino Flash Player 5.
- Questa appendice può essere consultata per ottenere le informazioni necessarie per la creazione degli script se già si utilizza ActionScript 1.0 per creare script orientati agli oggetti e non è ancora possibile passare ad ActionScript 2.0.

Se ActionScript non è mai stato utilizzato per creare script orientati agli oggetti e non è necessario impostare come destinazione Flash Player 5, non occorre consultare le informazioni presenti in questa appendice, in quanto la creazione di script tramite ActionScript 1.0 è sconsigliata. Al contrario, per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Questo capitolo contiene le seguenti sezioni:

<a href="#">Informazioni su ActionScript 1.0</a> .....	856
<a href="#">Creazione di un oggetto personalizzato in ActionScript 1.0</a> .....	858
<a href="#">Assegnazione di metodi a un oggetto personalizzato in ActionScript 1.0</a> .....	859
<a href="#">Definizione dei metodi del gestore di eventi in ActionScript 1.0</a> .....	860
<a href="#">Creazione dell'ereditarietà in ActionScript 1.0</a> .....	863
<a href="#">Aggiunta di proprietà getter/setter agli oggetti in ActionScript 1.0</a> .....	864
<a href="#">Uso delle proprietà dell'oggetto Function in ActionScript 1.0</a> .....	865



In alcuni esempi presenti in questa appendice viene utilizzato il metodo `Object.registerClass()`. Questo metodo è supportato solo in Flash Player 6 e nelle versioni successive. Non utilizzarlo se si imposta come destinazione Flash Player 5.

# Informazioni su ActionScript 1.0

## NOTA

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, "Classi" a pagina 197](#).

ActionScript è un linguaggio di programmazione orientato agli oggetti. Nella programmazione orientata agli oggetti sono utilizzati *oggetti* o strutture di dati, per raggruppare proprietà e metodi che consentono di controllare il comportamento o l'aspetto dell'oggetto. Gli oggetti consentono di organizzare e riutilizzare il codice. Una volta definito un oggetto, è possibile farvi riferimento per nome, senza doverlo ridefinire ogni volta che lo si utilizza.

Una *classe* costituisce una categoria generica di oggetti, che definisce una serie di oggetti dotati di proprietà comuni, controllabili con gli stessi metodi. Le proprietà sono attributi che definiscono un oggetto, ad esempio le dimensioni, la posizione, il colore, la trasparenza e così via. Le proprietà sono definite per una classe, mentre i valori delle proprietà vengono impostati per i singoli oggetti presenti nella classe. I metodi sono funzioni che consentono di impostare o richiamare le proprietà di un oggetto. Ad esempio, è possibile definire un metodo per calcolare le dimensioni di un oggetto. Analogamente alle proprietà, i metodi vengono definiti per una classe di oggetti e vengono quindi richiamati per i singoli oggetti della classe. ActionScript comprende numerose classi incorporate, tra cui MovieClip, Sound e altre. È possibile anche creare classi personalizzate per definire categorie di oggetti per le applicazioni. In ActionScript, gli oggetti possono essere semplici contenitori di dati oppure elementi grafici rappresentati sullo stage come clip filmato, pulsanti o campi di testo. Tutti i clip filmato sono istanze della classe incorporata MovieClip e tutti i pulsanti sono istanze della classe incorporata Button. Ogni istanza di un clip filmato contiene tutte le proprietà (ad esempio `_height`, `_rotation`, `_totalframes`) e tutti i metodi (ad esempio `gotoAndPlay()`, `loadMovie()`, `startDrag()`) della classe MovieClip.

Per definire una classe, è necessario creare una funzione speciale, denominata *funzione di costruzione*. Le classi incorporate sono dotate di funzioni di costruzione incorporate. Ad esempio, se si desidera ottenere informazioni relative a un ciclista, all'interno dell'applicazione, è possibile creare una funzione di costruzione, `Biker()`, dotata delle proprietà `time` e `distance` e del metodo `getSpeed()`, che indica la velocità del ciclista:

```
function Biker(t, d) {
 this.time = t;
 this.distance = d;
 this.getSpeed = function() {return this.time / this.distance;};
}
```

In questo esempio viene creata una funzione che necessita di due informazioni o *parametri* per eseguire l'azione: *t* e *d*. Quando si chiama la funzione per creare nuove istanze dell'oggetto, i parametri vengono passati alla funzione. Il codice seguente crea due istanze dell'oggetto Biker denominate emma e hamish e controlla la velocità dell'istanza emma con l'ausilio del metodo *getSpeed()* del codice ActionScript precedente:

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
trace(emma.getSpeed()); // Traccia 6
```

Nella programmazione orientata agli oggetti le classi possono ricevere proprietà e metodi da altre classi in base a un ordine specifico, detto *ereditarietà*. È possibile usare l'ereditarietà per aumentare o ridefinire le proprietà e i metodi di una classe. Una classe che eredita proprietà o metodi da un'altra classe è detta *sottoclasse*. Una classe che passa proprietà o metodi a un'altra classe è detta *superclasse*. Una classe può essere sia una sottoclasse che una superclasse.

Un oggetto è un tipo di dati complesso, contenente zero o più proprietà e metodi. Ogni proprietà, come una variabile, è provvista di un nome e di un valore. Le proprietà sono associate all'oggetto e contengono valori modificabili e recuperabili. Questi valori possono appartenere a qualsiasi tipo di dati: stringa, numero, booleano, oggetto, clip filmato o non definito. Le proprietà seguenti sono di tipi di dati diversi:

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 609;
customer.mcInstanceName._visible = true;
```

La proprietà di un oggetto può a sua volta essere un oggetto. Nella riga 4 dell'esempio precedente, *account* è una proprietà dell'oggetto *customer* e *currentRecord* è una proprietà dell'oggetto *account*. Il tipo di dati della proprietà *currentRecord* è numerico.

# Creazione di un oggetto personalizzato in ActionScript 1.0

**NOTA**

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, "Classi" a pagina 197](#).

Per creare un oggetto personalizzato, è necessario definire una funzione di costruzione, alla quale viene sempre assegnato lo stesso nome del tipo di oggetto che essa crea. È possibile utilizzare la parola chiave `this` all'interno del corpo della funzione di costruzione per fare riferimento all'oggetto creato dalla funzione. Quando si chiama una funzione di costruzione, Flash passa `this` alla funzione come parametro nascosto. Il codice seguente, ad esempio, corrisponde a una funzione di costruzione che crea un cerchio con la proprietà `radius`:

```
function Circle(radius) {
 this.radius = radius;
}
```

Dopo aver definito la funzione di costruzione, è necessario creare un'istanza dell'oggetto. Inserire l'operatore `new` prima del nome della funzione di costruzione e assegnare alla nuova istanza un nome di variabile. Ad esempio, nel codice seguente viene usato l'operatore `new` per creare un oggetto `Circle` con un raggio di 5 che viene assegnato alla variabile `myCircle`:

```
myCircle = new Circle(5);
```

**NOTA**

Un oggetto presenta la stessa area di validità della variabile alla quale è assegnato.

# Assegnazione di metodi a un oggetto personalizzato in ActionScript 1.0

## NOTA

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, "Classi" a pagina 197](#).

È possibile definire i metodi di un oggetto all'interno di una funzione di costruzione dell'oggetto stesso. Questa tecnica tuttavia è sconsigliata in quanto il metodo viene definito ogni volta che viene utilizzata la funzione di costruzione. L'esempio seguente crea i metodi `getArea()` e `getDiameter()` e traccia l'area e il diametro dell'istanza `myCircle` creata con un raggio impostato su 55:

```
function Circle(radius) {
 this.radius = radius;
 this.getArea = function(){
 return Math.PI * this.radius * this.radius;
 };
 this.getDiameter = function() {
 return 2 * this.radius;
 };
}
var myCircle = new Circle(55);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

Ogni funzione di costruzione presenta una proprietà `prototype` creata in modo automatico al momento della definizione della funzione. Tale proprietà indica i valori predefiniti delle proprietà degli oggetti creati con quella funzione. Inoltre, ogni istanza di un oggetto presenta una proprietà `__proto__` che fa riferimento alla proprietà `prototype` della funzione di costruzione che l'ha creata. Quindi, se si assegnano i metodi alla proprietà `prototype` di un oggetto, questi sono disponibili per qualsiasi nuova istanza di quel dato oggetto. È preferibile assegnare un metodo alla proprietà `prototype` della funzione di costruzione, poiché essa esiste in una posizione alla quale faranno riferimento le nuove istanze dell'oggetto o della classe. È possibile usare le proprietà `prototype` e `__proto__` per estendere gli oggetti, in modo da poter usare di nuovo il codice in script orientati agli oggetti. Per ulteriori informazioni, consultare ["Creazione dell'ereditarietà in ActionScript 1.0" a pagina 863](#).

La procedura seguente illustra come assegnare un metodo `getArea()` a un oggetto personalizzato `Circle`.

### Per assegnare un metodo a un oggetto personalizzato:

1. Definire la funzione di costruzione `Circle()`:

```
function Circle(radius) {
 this.radius = radius;
}
```

2. Definire il metodo `getArea()` dell'oggetto `Circle` che calcola l'area del cerchio.

Nell'esempio seguente, per definire il metodo `getArea()` e assegnare la proprietà `getArea` all'oggetto prototipo del cerchio, è possibile utilizzare un valore letterale di funzione:

```
Circle.prototype.getArea = function () {
 return Math.PI * this.radius * this.radius;
};
```

3. L'esempio seguente crea un'istanza dell'oggetto `Circle`:

```
var myCircle = new Circle(4);
```

4. Chiamare il metodo `getArea()` del nuovo oggetto `myCircle` con l'ausilio del codice seguente:

```
var myCircleArea = myCircle.getArea();
trace(myCircleArea); // Traccia 50.265...
```

Il metodo `getArea()` viene cercato nell'oggetto `myCircle`. Poiché tale oggetto non dispone di un metodo `getArea()`, questo viene cercato nell'oggetto prototipo `Circle.prototype`. Il metodo viene quindi trovato, chiamato e viene tracciato `myCircleArea`.

## Definizione dei metodi del gestore di eventi in ActionScript 1.0

### NOTA

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, "Classi" a pagina 197](#).

È possibile creare una classe di ActionScript per clip filmato e definire i metodi del gestore di eventi nell'oggetto prototipo di questa nuova classe. La definizione dei metodi nell'oggetto prototipo consente a tutte le istanze di questo simbolo di rispondere nello stesso modo agli eventi.

È inoltre possibile aggiungere i metodi del gestore di eventi `onClipEvent()` o `on()` a un'istanza individuale per fornire istruzioni univoche che vengono eseguite solo quando si verifica l'evento di quell'istanza. I metodi `onClipEvent()` e `on()` non hanno la precedenza sul metodo del gestore di eventi; entrambi gli eventi determinano l'esecuzione degli script. Se tuttavia si definiscono sia i metodi del gestore di eventi nell'oggetto prototipo che un metodo del gestore di eventi per un'istanza specifica, la definizione dell'istanza sostituisce la definizione del prototipo.

**Per definire un metodo del gestore di eventi in un oggetto prototipo dell'oggetto:**

1. Creare un simbolo di clip filmato e impostare l'ID di concatenamento su `theID`, selezionando il simbolo nel pannello Libreria e quindi scegliendo Concatenamento dal menu Opzioni del pannello.
2. Nel pannello Azioni (Finestra > Azioni), utilizzare l'istruzione `function` per definire una nuova classe, come nell'esempio seguente:

```
// Definisce una classe
function myClipClass() {}
```

Questa nuova classe viene assegnata a tutte le istanze del clip filmato che vengono aggiunte all'applicazione dalla linea temporale o con il metodo `attachMovie()` o `duplicateMovieClip()`. Se si desidera che questi clip filmato abbiano accesso ai metodi e alle proprietà dell'oggetto MovieClip incorporato, la nuova classe deve ereditare dalla classe MovieClip.

3. Immettere il codice, come nell'esempio seguente:

```
// Eredita dalla classe MovieClip
myClipClass.prototype = new MovieClip();
```

In questo modo, la classe `myClipClass` eredita tutte le proprietà e tutti i metodi della classe MovieClip.

4. Immettere il codice, come nell'esempio seguente, per definire i metodi del gestore di eventi per la nuova classe:

```
// Definisce i metodi del gestore di eventi per la classe myClipClass
myClipClass.prototype.onLoad = function() {trace("movie clip loaded");}
myClipClass.prototype.onEnterFrame = function() {trace("movie clip
entered frame");}
```

5. Scegliere Finestra > Libreria per aprire il pannello Libreria nel caso in cui non sia già visualizzato.
6. Selezionare i simboli che si desidera associare alla nuova classe e scegliere Concatenamento dal menu a comparsa del pannello Libreria.

7. Nella finestra di dialogo Proprietà del concatenamento, selezionare Esporta per ActionScript.
8. Immettere un identificatore di concatenamento nella casella di testo Identificatore. L'identificatore di concatenamento deve essere lo stesso per tutti i simboli che si desidera associare alla nuova classe. Nell'esempio `myClipClass`, l'identificatore è `theID`.
9. Immettere il codice, come nell'esempio seguente, nel pannello Azioni:

```
// Registra la classe
Object.registerClass("theID", myClipClass);
this.attachMovie("theID","myName",1);
```

In questo modo, il simbolo con identificatore di concatenamento `theID` viene registrato con la classe `myClipClass`. Tutte le istanze di `myClipClass` dispongono di metodi del gestore di eventi con lo stesso comportamento definito al punto 4, che, a sua volta, equivale a quello di tutte le istanze della classe `MovieClip`, perché al punto 3 la nuova classe eredita dalla classe `MovieClip`.

Il codice completo è riportato nell'esempio seguente:

```
function myClipClass(){}
myClipClass.prototype = new MovieClip();
myClipClass.prototype.onLoad = function(){
 trace("movie clip loaded");
}
myClipClass.prototype.onPress = function(){
 trace("pressed");
}

myClipClass.prototype.onEnterFrame = function(){
 trace("movie clip entered frame");
}

myClipClass.prototype.myfunction = function(){
 trace("myfunction called");
}

Object.registerClass("myclipID",myClipClass);
this.attachMovie("myclipID","clipName",3);
```

# Creazione dell'ereditarietà in ActionScript 1.0

## NOTA

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, "Classi" a pagina 197](#).

L'ereditarietà consente di organizzare, estendere e riutilizzare le funzionalità. Le sottoclassi ereditano le proprietà e i metodi dalle superclassi aggiungendo le proprie proprietà e i propri metodi specializzati. Ad esempio, facendo riferimento al mondo reale, Bike (Bicicletta) è una superclasse e MountainBike e Tricycle (Triciclo) sono sottoclassi di Bike. Entrambe le sottoclassi contengono o *ereditano* i metodi e le proprietà della superclasse, ad esempio ruote. Ogni sottoclasse presenta inoltre delle proprietà e dei metodi propri che estendono la superclasse. Ad esempio, la sottoclasse MountainBike presenta una proprietà cambio. In ActionScript, è possibile usare gli elementi prototype e \_\_proto\_\_ per rendere possibile l'ereditarietà.

Tutte le funzioni dispongono di una proprietà prototype che viene creata automaticamente al momento della definizione della funzione. Tale proprietà indica i valori predefiniti delle proprietà degli oggetti creati con quella funzione. È possibile usare la proprietà prototype per assegnare proprietà e metodi a una classe. Per ulteriori informazioni, consultare ["Assegnazione di metodi a un oggetto personalizzato in ActionScript 1.0" a pagina 859](#).

Tutte le istanze di una classe dispongono di una proprietà \_\_proto\_\_ che indica da quale oggetto la classe eredita. Quando si usa una funzione di costruzione per creare un oggetto, la proprietà \_\_proto\_\_ viene impostata per creare un riferimento alla proprietà prototype della funzione di costruzione.

L'ereditarietà si basa su una gerarchia precisa. Quando si chiama una proprietà o un metodo di un oggetto, ActionScript verifica se tale elemento esiste nell'oggetto. Se l'elemento non esiste, l'informazione viene cercata nella proprietà \_\_proto\_\_ (myObject.\_\_proto\_\_). Se la proprietà chiamata non è una proprietà dell'oggetto \_\_proto\_\_, la ricerca viene eseguita in myObject.\_\_proto\_\_.\_\_proto\_\_ e così via.

Il seguente esempio definisce la funzione di costruzione Bike():

```
function Bike(length, color) {
 this.length = length;
 this.color = color;
 this.pos = 0;
}
```

Il codice seguente aggiunge il metodo roll() alla classe Bike:

```
Bike.prototype.roll = function() {return this.pos += 20;};
```

È quindi possibile tracciare la posizione della bicicletta con il codice seguente:

```
var myBike = new Bike(55, "blue");
trace(myBike.roll()); // Traccia 20.
trace(myBike.roll()); // Traccia 40.
```

Aniché aggiungere `roll()` alle classi `MountainBike` e `Tricycle`, è possibile creare la classe `MountainBike` utilizzando `Bike` come superclasse, come nell'esempio seguente:

```
MountainBike.prototype = new Bike();
```

È ora possibile chiamare il metodo `roll()` di `MountainBike`, come nell'esempio seguente:

```
var myKona = new MountainBike(20, "teal");
trace(myKona.roll()); // Traccia 20.
```

I clip filmato non ereditano elementi l'uno dall'altro. Per consentire l'ereditarietà tra clip filmato, è possibile usare il metodo `Object.registerClass()` per assegnare ai clip filmato una classe diversa dalla classe `MovieClip`.

## Aggiunta di proprietà getter/setter agli oggetti in ActionScript 1.0



Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il Capitolo 6, “Classi” a pagina 197.

È possibile creare proprietà getter/setter per un oggetto utilizzando il metodo `Object.addProperty()`.

Una *funzione getter* non prevede l'uso di parametri, può restituire qualsiasi tipo di valore e può cambiare tipo tra una chiamata e un'altra. Il valore restituito viene considerato come valore corrente della proprietà.

Una *funzione setter* è una funzione che assume un parametro come nuovo valore della proprietà. Ad esempio, se la proprietà `x` viene assegnata mediante l'istruzione `x = 1`, alla funzione setter viene passato il parametro `1` di tipo numerico. Il valore restituito dalla funzione setter viene ignorato.

Quando Flash legge una proprietà getter/setter, invoca la funzione getter e il valore restituito dalla funzione diventa un valore di `prop`. Quando Flash scrive una proprietà getter/setter, richiama la funzione setter e passa a questa il nuovo valore come parametro. Se esiste già una proprietà con lo stesso nome assegnato alla nuova proprietà, quest'ultima la sovrascrive.

È possibile aggiungere proprietà getter/setter agli oggetti prototipo. In tal caso, tutte le istanze associate all'oggetto che ereditano l'oggetto prototipo ereditano anche la proprietà getter/setter. È possibile aggiungere una proprietà getter/setter in un'unica posizione, ovvero nell'oggetto prototipo, e propagarla a tutte le istanze di una classe, in modo simile a come vengono aggiunti i metodi agli oggetti prototipo. Se viene richiamata una funzione getter/setter per una proprietà getter/setter in un oggetto prototipo ereditato, il riferimento passato alla funzione getter/setter corrisponde all'oggetto a cui si era fatto riferimento in origine e non all'oggetto prototipo.

In modalità di prova il comando Debug > Elenco variabili supporta le proprietà getter/setter che è possibile aggiungere agli oggetti tramite il metodo `Object.addProperty()`. Le proprietà aggiunte a un oggetto in questo modo vengono visualizzate con le altre proprietà dell'oggetto nel pannello Output. Nel pannello Output le proprietà getter/setter sono identificate con il prefisso `[getter/setter]`. Per ulteriori informazioni sul comando Elenco variabili, consultare “[Uso del pannello Output](#)” a pagina 785.

## Uso delle proprietà dell'oggetto Function in ActionScript 1.0



Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, “Classi” a pagina 197](#).

È possibile specificare l'oggetto a cui viene applicata una funzione e i valori di parametro che vengono passati alla funzione utilizzando i metodi `call()` e `apply()` dell'oggetto Function. Ogni funzione di ActionScript è rappresentata da un oggetto Function, pertanto tutte le funzioni supportano i metodi `call()` e `apply()`. Quando si utilizza una funzione di costruzione per la creazione di una classe personalizzata o si utilizza una funzione per definire i metodi per una classe personalizzata, è possibile richiamare i metodi `call()` e `apply()` per la funzione.

## Uso del metodo Function.call() per invocare una funzione in ActionScript 1.0

**NOTA**

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Il metodo `Function.call()` richiama la funzione rappresentata da un oggetto `Function`.

In quasi tutti i casi, è possibile utilizzare l'operatore di chiamata della funzione `( )` invece del metodo `call()`. L'operatore di chiamata della funzione crea un codice conciso e leggibile. Il metodo `call()` risulta utile soprattutto quando il parametro `this` della chiamata della funzione deve essere esplicitamente controllato. In genere, se una funzione viene richiamata come metodo di un oggetto all'interno del corpo della funzione, `this` viene impostato su `myObject`, come nell'esempio seguente:

```
myObject.myMethod(1, 2, 3);
```

In alcuni casi, può essere necessario far sì che `this` punti a un'altra posizione, ad esempio se è necessario richiamare una funzione come metodo di un oggetto quando la funzione non è effettivamente memorizzata come metodo dell'oggetto, come illustrato nell'esempio seguente:

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

È possibile passare il valore `null` per il parametro `thisObject` per richiamare una funzione come funzione standard e non come metodo di un oggetto. Le seguenti chiamate di funzione, ad esempio, sono equivalenti:

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

### Per richiamare una funzione tramite il metodo `Function.call()`:

- Utilizzare la sintassi seguente:

```
myFunction.call(thisObject, parameter1, ..., parameterN)
```

Il metodo acquisisce i parametri seguenti:

- Il parametro `thisObject` specifica il valore di `this` all'interno del corpo della funzione.
- I parametri `parameter1`..., `parameterN` indicano i parametri da passare a `myFunction`. È possibile specificare zero o più parametri.

## Definizione dell'oggetto a cui la funzione viene applicata utilizzando Function.apply() in ActionScript 1.0

**NOTA**

Molti utenti di Flash possono trarre grandi vantaggi dall'uso di ActionScript 2.0, specialmente nel caso di applicazioni complesse. Per informazioni sull'uso di ActionScript 2.0, consultare il [Capitolo 6, “Classi” a pagina 197](#).

Il metodo `Function.apply()` specifica il valore di `this` da utilizzare all'interno delle funzioni chiamate da ActionScript. Questo metodo specifica inoltre i parametri da passare a ogni funzione chiamata.

I parametri vengono specificati come oggetto `Array`. Ciò risulta spesso utile quando non si conosce il numero dei parametri da passare finché lo script non viene effettivamente eseguito.

Per ulteriori informazioni, consultare `apply` (metodo `Function.apply`) nella *Guida di riferimento di ActionScript 2.0*.

### Per specificare l'oggetto a cui viene applicata una funzione tramite `Function.apply()`:

- Utilizzare la sintassi seguente:

```
myFunction.apply(thisObject, argumentsObject)
```

Il metodo acquisisce i parametri seguenti:

- Il parametro `thisObject` specifica l'oggetto a cui viene applicata la funzione `myFunction`.
- Il parametro `argumentsObject` definisce un array i cui elementi vengono passati alla funzione `myFunction` come parametri.



# Terminologia

F

Come qualsiasi altro linguaggio per la creazione di script, ActionScript utilizza una terminologia propria. Macromedia Flash usa anche molta terminologia univoca. L'elenco seguente fornisce un'introduzione ai termini più importanti di ActionScript e ai termini di Flash relativi alla programmazione con ActionScript che sono tipici dell'utilizzo dell'ambiente di creazione Flash.

**Editor di ActionScript:** l'editor di codice nel pannello Azioni e nella finestra Script. L'editor di ActionScript dispone di numerose funzionalità, come la formattazione automatica, la visualizzazione dei caratteri nascosti e la codifica con colori di parti degli script. (Vedere anche: finestra Script, pannello Azioni).

**Pannello Azioni:** è il pannello nell'ambiente di creazione Flash in cui si scrive il codice ActionScript.

**Funzione anonima:** una funzione senza nome che fa riferimento a se stessa; si fa riferimento alla funzione anonima quando la si crea. Per informazioni e un esempio, vedere “[Scrittura di funzioni anonime e di callback](#)” a pagina 176.

**Alias:** un testo reso mediante aliasing che non usa variazioni di colore per smussare i contorni irregolari, a differenza di un testo con anti-aliasing (vedere la definizione seguente).

**Anti-alias:** caratteri con anti-aliasing per smussare il testo in modo che i bordi dei caratteri visualizzati sullo schermo appaiano meno irregolari. L'opzione Antialiasing in Flash allinea i contorni del testo lungo i limiti di pixel in modo da rendere il testo più leggibile ed è particolarmente efficace per il rendering dei caratteri di piccole dimensioni.

**Array:** oggetti le cui proprietà sono identificate da un numero che ne rappresenta la posizione nella struttura dell'array. Un array è fondamentalmente un elenco di elementi.

**Ambiente di creazione:** l'area di lavoro Flash che comprende tutti gli elementi dell'interfaccia utente. Con l'ambiente di creazione si creano file FLA o script (nella finestra Script).

**Immagini bitmap** (o grafici *raster*): generalmente immagini fotografiche o grafici con numerosi dettagli. Ogni pixel (o *bit*) dell'immagine contiene un dato e, insieme, questi bit formano l'immagine stessa. Le bitmap possono essere salvate nei formati JPEG, BMP o GIF. Un altro tipo di grafico, diverso dalle bitmap, è il formato *vettoriale*.

**Valore booleano:** valore true o false.

**Memorizzazione nella cache:** le informazioni che vengono riutilizzate nell'applicazione o che sono memorizzate sul computer per poter essere riutilizzate. Ad esempio, se si scarica un'immagine da Internet, spesso viene memorizzata nella cache per poterla visualizzare in un momento successivo senza dovere scaricare di nuovo i dati.

**Funzioni di callback:** funzioni anonime che vengono associate a un determinato evento. Una funzione chiama una funzione di callback dopo che si verifica un evento specifico, ad esempio dopo il termine del caricamento (`onLoad()`) o dell'animazione di un oggetto (`onMotionFinished()`). Per ulteriori informazioni e un esempio, vedere “[Scrittura di funzioni anonime e di callback](#)” a pagina 176.

**Caratteri :** lettere, numeri e simboli di punteggiatura che vengono combinati per creare stringhe. Sono detti anche *glifi*.

**Classi:** tipi di dati che è possibile creare per definire un nuovo tipo di oggetto. Per definire una classe, è necessario usare la parola chiave `class` in un file di script esterno (non in uno script creato nel pannello Azioni).

**Percorso di classe:** l'elenco di cartelle nelle quali Flash esegue la ricerca delle classi o delle definizioni di interfaccia. Quando si crea un file di classe, è necessario salvare il file in una delle directory specificate nel percorso di classe o in una relativa sottodirectory. I percorsi di classe si trovano sia al livello globale (applicazione) che al livello di documento.

**Costanti:** elementi che non cambiano valore. Ad esempio, la costante `Key.TAB` indica sempre lo stesso elemento, ossia il tasto TAB presente sulla tastiera. Le costanti sono utili per eseguire confronti tra valori.

**Funzioni di costruzione** (o *costruttori*): funzioni che consentono di definire (inizializzare) le proprietà e i metodi di una classe. Per definizione, le funzioni di costruzione, o costruttori, sono funzioni incluse in una definizione di classe e hanno lo stesso nome della classe. Nel codice seguente, ad esempio, viene definita una classe `Circle` e viene implementata una funzione di costruzione:

```
// File Circle.as
class Circle {
 private var circumference:Number;
 // funzione di costruzione
 function Circle(radius:Number){
 this.circumference = 2 * Math.PI * radius;
 }
}
```

Il termine *funzione di costruzione* viene inoltre utilizzato quando si crea un oggetto (ovvero se ne crea un'istanza) in base a una classe specifica. Le istruzioni seguenti sono delle chiamate a funzioni di costruzione della classe Array incorporata e della classe personalizzata Circle:

```
var my_array:Array = new Array();
var my_circle:Circle = new Circle(9);
```

**Tipi di dati:** tipo di informazioni che le variabili o gli elementi di ActionScript possono contenere. I tipi di dati incorporati di ActionScript sono: stringa, numerico, booleano, oggetto, MovieClip, Function, null e undefined. Per ulteriori informazioni, vedere “[Informazioni sui tipi di dati](#)” a pagina 334.

**Caratteri dispositivo:** caratteri speciali in Flash che non sono incorporati in un file SWF. Flash Player utilizza invece i caratteri disponibili sul computer locale che più assomigliano ai caratteri dispositivo. Poiché i profili dei caratteri non sono incorporati, la dimensione del file SWF è minore di quando vengono utilizzati i profili dei caratteri incorporati. Tuttavia, poiché i caratteri dispositivo non sono incorporati, se sul sistema non è installato un tipo di carattere corrispondente al carattere dispositivo, l'aspetto del testo potrebbe risultare diverso da quello previsto. Flash include tre tipi di carattere dispositivo: \_sans (simile a Helvetica o Arial), \_serif (simile a Times Roman) e \_typewriter (simile a Courier).

**Sintassi del punto:** si utilizza un operatore punto (.) per accedere alle proprietà o ai metodi che appartengono a un oggetto o a un'istanza sullo stage con ActionScript. L'operatore punto consente inoltre di identificare il percorso target di un'istanza (ad esempio un clip filmato), di una variabile, di una funzione o di un oggetto. Un'espressione in cui è utilizzato questo tipo di sintassi inizia con il nome dell'oggetto o del clip filmato seguito da un punto e termina con l'elemento che si desidera specificare.

**Eventi:** azioni che si verificano durante la riproduzione di un file SWF. Eventi diversi vengono generati, ad esempio, quando si carica un clip filmato, quando l'indicatore di riproduzione raggiunge un fotogramma, quando l'utente seleziona un pulsante o un clip filmato oppure digita sulla tastiera.

**Gestori di eventi:** azioni speciali che consentono di gestire eventi quando si fa clic con il mouse o quando si è concluso un caricamento di dati. In ActionScript sono disponibili due tipi di gestori di eventi: metodi del gestore di eventi e listener di eventi. Esistono inoltre due gestori di eventi, on handler e onClipEvent handler, che è possibile assegnare direttamente a pulsanti e clip filmato. Nella casella degli strumenti Azioni, ciascun oggetto ActionScript che dispone di metodi del gestore di eventi o listener di eventi presenta una sottocategoria denominata Eventi o Listener. Alcuni comandi possono essere utilizzati sia come gestori di eventi che come listener di eventi e sono inclusi in entrambe le sottocategorie. Per ulteriori informazioni sulla gestione di eventi, vedere “[Gestione degli eventi](#)” a pagina 311.

**Espressioni:** combinazioni valide di simboli ActionScript che rappresentano un valore. Un'espressione è costituita da operatori e operandi. Ad esempio, nell'espressione `x + 2`, `x` e `2` sono operandi e `+` è un operatore.

**Contenitore di Flash Player:** il sistema che contiene l'applicazione Flash, come un browser o l'applicazione desktop. È possibile aggiungere ActionScript e JavaScript per semplificare le comunicazioni tra il contenitore di Flash Player e un file SWF.

**FlashType:** la tecnologia migliorata di rendering del testo in Flash 8. Ad esempio, Aliasing per leggibilità utilizza la tecnologia di rendering FlashType, al contrario di Aliasing per animazione. Per ulteriori informazioni, vedere “[Informazioni sul rendering dei caratteri e sul testo con antialiasing](#)” a pagina 443.

**Script di fotogramma:** blocchi di codice da aggiungere a un fotogramma su una linea temporale.

**Funzioni:** blocchi di codice riutilizzabili a cui è possibile passare parametri e che possono restituire un valore. Per ulteriori informazioni, vedere “[Informazioni su funzioni e metodi](#)” a pagina 169.

**Valore letterale di funzione:** una funzione senza nome dichiarata in un'espressione anziché in un'istruzione. I valori letterali di funzione consentono di utilizzare una funzione temporaneamente o al posto di un'espressione.

**IDE:** acronimo per “ambiente di sviluppo integrato”, cioè un'applicazione nella quale lo sviluppatore può scrivere, provare ed eseguire il debug di applicazioni in un ambiente interattivo. Lo strumento di creazione di Flash viene talvolta chiamato un IDE.

**Identifieri:** nomi usati per indicare una variabile, una proprietà, un oggetto, una funzione o un metodo. Il primo carattere deve essere costituito da una lettera, un carattere di sottolineatura (`_`) o dal simbolo del dollaro (`$`). Ogni carattere successivo deve essere una lettera, un numero, un carattere di sottolineatura o il simbolo del dollaro. Ad esempio, `firstName` è il nome di una variabile.

**Istanze:** oggetti che contengono tutte le proprietà e i metodi di una determinata classe. Tutti gli array, ad esempio, sono istanze della classe Array ed è pertanto possibile usare qualsiasi metodo o proprietà della classe Array con qualsiasi istanza di array.

**Nomi di istanze:** nomi univoci che consentono di fare riferimento alle istanze create o alle istanze di un clip filmato e dei pulsanti sullo stage. Nel codice seguente, ad esempio, `names` e `studentName` sono nomi di istanze di due oggetti, un array e una stringa:

```
var names:Array = new Array();
var studentName:String = new String();
```

È possibile usare la finestra di ispezione Proprietà per assegnare i nomi alle istanze presenti sullo stage. Un simbolo principale contenuto nella libreria, ad esempio, può essere denominato counter e le due istanze dello stesso simbolo nel file SWF possono avere nomi di istanze scorePlayer1\_mc e scorePlayer2\_mc. Nel codice seguente viene impostata una variabile denominata score all'interno di ciascuna istanza di clip filmato utilizzando i nomi delle istanze:

```
this.scorePlayer1_mc.score = 0;
this.scorePlayer2_mc.score = 0;
```

Per creare istanze è possibile utilizzare la tipizzazione forte dei dati affinché durante la digitazione del codice vengano visualizzati suggerimenti sul codice.

**Parole chiave:** parole riservate che hanno un significato speciale. Ad esempio, var è una parola chiave usata per dichiarare le variabili locali. Non è possibile usare una parola chiave come identificatore. Ad esempio, var non è un nome di variabile consentito. Per un elenco delle parole chiave, vedere “[Informazioni sulle parole chiave](#)” a pagina 103 e “[Informazioni sulle parole riservate](#)” a pagina 104.

**Valori letterali:** valori che hanno un tipo particolare, come valori letterali numerici o valori letterali di stringa. I valori letterali non vengono memorizzati in una variabile. Un valore letterale è un valore che compare direttamente nel codice ed è un valore costante (invariabile) nei documenti Flash. Vedere anche *funzione letterale stringa letterale*.

**Metodi:** funzioni associate a una classe. sortOn(), ad esempio, è un metodo incorporato associato alla classe Array. È inoltre possibile creare funzioni che operano in modo analogo ai metodi per gli oggetti basati su classi incorporate o su classi create dall'utente. Ad esempio, nel codice seguente, clear() diventa un metodo di un oggetto controller definito in precedenza:

```
function reset(){
 this.x_pos = 0;
 this.y_pos = 0;
}
controller.clear = reset;
controller.clear();
```

Negli esempi seguenti viene illustrato come creare metodi di una classe:

```
//Esempio ActionScript 1.0
A = new Object();
A.prototype.myMethod = function() {
 trace("myMethod");
}

//Esempio ActionScript 2.0
class B {
 function myMethod() {
 trace("myMethod");
 }
}
```

**Funzione con nome:** un tipo di funzione che viene comunemente creata nel codice ActionScript per eseguire tutti i tipi di azioni. Per informazioni e un esempio, vedere “[Scrittura di funzioni con nome](#)” a pagina 175.

**Codice di un oggetto:** il codice ActionScript associato alle istanze. Per aggiungere il codice di un oggetto, si seleziona un'istanza sullo stage e si inserisce il codice nel pannello Azioni. Non è consigliabile associare codice a oggetti sullo stage. Per informazioni sulle procedure consigliate, vedere “[Procedure ottimali e convenzioni di codifica per ActionScript 2.0](#)” a pagina 791.

**Oggetti:** insiemi di proprietà e di metodi; ogni oggetto presenta un nome e rappresenta un'istanza di una determinata classe. Nel linguaggio ActionScript gli oggetti incorporati sono predefiniti. La classe incorporata Date, ad esempio, fornisce informazioni provenienti dall'orologio di sistema.

**Operatori:** termini che calcolano un nuovo valore in base a uno o più valori. L'operatore di addizione (+), ad esempio, consente di sommare due o più valori per creare un nuovo valore. I valori gestiti dagli operatori sono denominati *operandi*.

**Parametri:** (denominati anche *argomenti*) segnaposto che consentono di passare i valori alle funzioni. Nella funzione `welcome()` riportata di seguito, ad esempio, vengono utilizzati due valori ricevuti nei parametri `firstName` e `hobby`:

```
function welcome(firstName:String, hobby:String):String {
 var welcomeText:String = "Hello, " + firstName + ". I see you enjoy " +
 hobby +".";
 return welcomeText;
}
```

**Pacchetti:** directory che contengono uno o più file di classe e che risiedono nella directory del percorso di classe definita (vedere “[Informazioni sui pacchetti](#)” a pagina 200).

**Blocco degli script:** consente di bloccare più script da diversi oggetti e lavorare contemporaneamente con essi nel pannello Azioni. Si consiglia di utilizzare questa funzione con Script navigator.

**JPEG a scaricamento progressivo:** immagini che vengono costruite e visualizzate gradualmente durante lo scarico da un server. Una normale immagine JPEG viene visualizzata riga per riga durante lo scarico da un server.

**Proprietà:** attributi che definiscono un oggetto. `length`, ad esempio, è una proprietà di tutti gli array che specifica il numero di elementi nell'array.

**Segni di punteggiatura:** caratteri speciali che sono di aiuto durante la scrittura di codice ActionScript. In Flash sono disponibili molti segni di punteggiatura. I più comuni sono il punto e virgola (;), i due punti (:), le parentesi [] e le parentesi graffe {}. Ognuno di essi ha un significato particolare nel linguaggio di Flash e consente di definire i tipi di dati, di terminare le istruzioni o la struttura di ActionScript.

**Assistente script:** la nuova modalità Assistente script del pannello Azioni. Assistente script consente una maggiore facilità di creazione degli script anche agli utenti che non conoscono ActionScript in modo approfondito. Assistente script aiuta l'utente nella creazione degli script mediante la selezione delle voci nella casella degli strumenti Azioni del pannello Azioni e grazie a un'interfaccia di campi di testo, pulsanti di opzione e caselle di controllo che chiedono l'immissione di variabili esatte e altri costrutti del linguaggio di script. Questa funzionalità è simile alla *modalità normale* delle precedenti versioni dello strumento di creazione Flash.

**Riquadro dello script:** il riquadro nel pannello Azioni o nella finestra Script; cioè l'area in cui si scrive il codice ActionScript.

**Finestra Script:** l'ambiente di modifica in cui si creano e si modificano script esterni, come file JavaScript Flash o file ActionScript. Ad esempio, selezionare File > Nuovo e quindi File ActionScript per usare la finestra Script per scrivere un file di classe.

**Istruzioni:** elementi del linguaggio che eseguono o specificano un'azione. L'istruzione `return`, ad esempio, restituisce un risultato sotto forma di valore della funzione in cui l'istruzione viene eseguita. L'istruzione `if` valuta una condizione per determinare l'azione successiva da eseguire. L'istruzione `switch` crea una struttura ad albero per le istruzioni ActionScript.

**Stringa:** una sequenza di caratteri e un tipo di dati. Per ulteriori informazioni, vedere “[Informazioni sulle stringhe e sulla classe String](#)” a pagina 492.

**Valore letterale di stringa:** una sequenza di caratteri delimitata da virgolette semplici diritte. I caratteri sono i dati stessi, non un riferimento ai dati. Un valore letterale di stringa non è un oggetto Stringa. Per ulteriori informazioni, vedere “[Informazioni sulle stringhe e sulla classe String](#)” a pagina 492.

**Superficie:** clip filmato per cui è stato attivato l'indicatore di memorizzazione delle bitmap nella cache. Per informazioni sull'uso del caching, vedere “[Memorizzazione di un clip filmato nella cache](#)” a pagina 406.

**Sintassi:** la grammatica e l'ortografia del linguaggio utilizzato per la programmazione. In caso di sintassi non corretta, il compilatore non è in grado di comprendere il codice, pertanto vengono visualizzati errori o avvisi nel pannello Output quando si tenta di provare il documento nell'ambiente di prova. La sintassi può quindi essere definita come un insieme di regole e linee guida che aiutano a creare codice ActionScript corretto.

**Percorsi target:** indirizzi gerarchici dei nomi di istanze dei clip filmato, delle variabili e degli oggetti di un file SWF. È possibile assegnare un nome all'istanza di un clip filmato nella finestra di ispezione Proprietà. (La linea temporale principale è sempre denominata \_root.) Un percorso target consente di indirizzare un'azione a un clip filmato oppure per ottenere o impostare il valore di una variabile o di una proprietà. L'istruzione seguente rappresenta, ad esempio, il percorso target della proprietà volume dell'oggetto denominato stereoControl:

```
stereoControl.volume
```

**Testo:** una serie di una o più stringhe che può essere visualizzata in un campo di testo o in un componente dell'interfaccia utente.

**Campi di testo:** elementi grafici sullo stage che permettono di visualizzare un testo per l'utente e che possono essere creati con lo strumento Testo o mediante un codice ActionScript. Flash permette di impostare campi di testo modificabili (sola lettura), con formattazione HTML, con il supporto multiriga, effetto maschera con password o applicare un foglio di stile CSS al testo formattato in HTML.

**Formattazione del testo:** può essere applicata a un campo di testo o ad alcuni caratteri di un campo di testo. Alcuni esempi di opzioni di formattazione del testo sono: allineamento, rientri, grassetto, colore, dimensione dei caratteri, larghezza margini, corsivo e spaziatura.

**Funzioni di primo livello:** funzioni che non appartengono a una classe (talvolta chiamate funzioni *predefined* o *incorporate*), in quanto possono essere chiamate senza una funzione di costruzione. Esempi di funzioni incorporate nel primo livello del linguaggio ActionScript sono `trace()` e `setInterval()`.

**Funzioni definite dall'utente:** funzioni create dall'utente per essere utilizzate nelle applicazioni. Si contrappongono alle funzioni delle classi incorporate che invece eseguono operazioni predefinite. Il nome della funzione viene assegnato dal programmatore che aggiunge quindi le istruzioni all'interno del blocco della funzione.

**Variabili:** identificatori che contengono i valori di qualsiasi tipo di dati. Le variabili possono essere create, modificate e aggiornate. È possibile recuperare i valori in esse contenuti e usarli negli script. Nell'esempio seguente gli identificatori a sinistra del segno di uguale sono variabili:

```
var x:Number = 5;
var name:String = "Lolo";
var c_color:Color = new Color(mcinstanceName);
```

Per ulteriori informazioni sulle variabili, vedere “[Informazioni sulle variabili](#)” a pagina 350.

**Immagini vettoriali:** descrivono le immagini utilizzando linee e curve, denominate vettori, che includono anche proprietà relative al colore e alla posizione. Ogni vettore utilizza calcoli matematici e non bit per descrivere la forma, consentendone la modifica in scala senza degrado nella qualità. Un altro tipo di grafico è la *bitmap*, che è rappresentata da punti o pixel.

# Indice analitico

## Simboli

\\" 503  
\' 503  
\b 503  
\f 503  
\n 503  
\r 503  
\t 503  
\unnnn 503  
\xnn 503  
\_lockroot, uso 810

## A

ActionScript  
confronto tra le versioni 73  
creazione di cue point con 671  
Flash Player 824  
formattazione 54  
impostazioni di pubblicazione 67  
informazioni 71, 72  
modifica delle preferenze 45  
ActionScript 2.0  
assegnazione della classe ActionScript 2.0 ai clip  
filmato 412  
messaggi di errore del compilatore 835  
ActiveX, controlli 723  
ADF 446, 449  
alias, definizione 869  
ambiente di creazione 869  
andamento  
definizione 527  
informazioni 534  
mediante codice 537  
animazione  
brillantezza 574

creazione di una barra di avanzamento 681  
filtri 582  
frequenza fotogrammi 515, 539  
mediante filtro bagliore 545  
animazione con script  
API di disegno 604  
classi Tween e TransitionManager 527  
creazione di una barra di avanzamento 681  
e classe Tween 582  
e filtri 582  
e filtro sfocatura 582  
informazioni 514  
interpolazione luminosità 521  
panoramica di immagini 524  
spostamento degli oggetti 522  
spostamento di immagini 524  
animazione, simboli e 339  
animazioni  
continuazione 540  
eseguite in modo continuo 541  
anti-aliasing  
definizione 869  
antialiasing  
per animazione e leggibilità 445  
antialiasing personalizzato  
definizione 445  
antiAliasType, proprietà 447, 450, 453  
API di disegno  
barra di avanzamento 692  
disegno di cerchi 595  
disegno di curve 592  
disegno di forme specifiche 590, 593  
disegno di linee, curve e forme 591  
disegno di rettangoli 593  
disegno di rettangoli arrotondati 594  
disegno di triangoli 592, 596  
e stili di linee 598

informazioni 590  
linee e riempimenti 630  
riempimenti con gradiente complessi 597  
uso 692

API External  
informazioni 723  
uso 724

applicazione dell'effetto maschera ai canali alfa 411  
applicazione della dissolvenza agli oggetti 516  
applicazioni Web, connessione continua 717  
architettura basata su componenti, definizione 381  
area di validità  
informazioni 86  
nelle classi 811  
procedure ottimali 809  
this, parola chiave 329

area di validità di \_root 86

ARGB (RGB con Alfa) 573

argomenti  
definizione 874  
nelle funzioni con nome 176  
*Vedere* parametri

array  
aggiunta e rimozione di elementi 134  
analogia 129  
array associativo 140  
array associativo tramite la funzione di costruzione di Array 142  
array associativo utilizzando Object 141  
array multidimensionali 136  
assegnazione di valori 354  
associativi 139  
creare un oggetto 377  
creazione 354  
e classe Object 143  
e metodo sortOn() 193  
elaborazione 130, 132  
elementi di 130  
esempi di 129, 131  
indicizzati 135  
informazioni 129  
iterazione in un array multidimensionali 138  
multidimensionali 136  
multidimensionali utilizzando un ciclo for 137  
passaggio per riferimento 361  
riferimento e determinazione della lunghezza 133  
sintassi abbreviata 129  
uso 130  
utilizzo della sintassi abbreviata per la creazione 355

array associativo, informazioni su 139

array con indice 131, 135  
array letterali 135  
array multidimensionali, informazioni su 136  
ASCII, definizione 493  
ASCII, valori 621  
altri tasti 848  
tasti del tastierino numerico 846  
tasti della tastiera 844  
tasti funzione 847

asincrone, azioni 691

ASO, file 258  
eliminazione 259  
uso 258

assegnazione di nomi a classi e oggetti, procedure consigliate 800

assegnazione di nomi ai pacchetti, procedure consigliate 802

assegnazione di nomi alle interfacce, procedure consigliate 803

associatività, degli operatori 146

associazione di audio 626

associazione di componenti con ActionScript 640

associazione di dati in fase di runtime  
creazione di un'associazione bidirezionale 635  
informazioni 632  
mediante CheckBox 636

associazione di dati, mediante ActionScript 632

associazioni  
creazione di un'associazione bidirezionale 635  
creazione di un'associazione unidirezionale 633  
creazione mediante AvtionScript 632

attivazione del debug remoto 775

audio  
aggiunta alla linea temporale 626  
controllo 625  
controllo del bilanciamento 627

Azioni, pannello  
casella degli strumenti Azioni 37  
definizione 869  
informazioni 36, 37  
menu a comparsa 43  
riquadro dello script 38  
Script navigator 37  
scrittura di codice 40

azioni, standard di codifica 807

## B

barra di avanzamento

creazione con il codice 681  
e API di disegno 692  
per caricamento dati 692  
barra, sintassi  
  informazioni 87  
  non supportata in ActionScript 2.0 87  
  uso 854  
bilanciamento (audio), controllo 627  
bilanciamento della punteggiatura, controllo 60  
bitmap  
  grafica 869  
  testo 445  
BitmapData, classe  
  effetto di disturbo 585  
  informazioni 584  
  mediante filtro mappa di spostamento 586  
  uso 585, 658  
blocco degli script  
  definizione 874  
  sulla linea temporale 64  
blocco di uno script 64  
Boolean  
  tipo di dati 337  
  valori 870

## C

cacheAsBitmap, proprietà 402  
caching, definizione 870  
campi con distanza a campionamento adattivo 449  
campi di testo  
  applicazione di CSS 466  
  applicazione di un flusso al testo intorno a immagini  
    incorporate 475, 479  
  caricamento di testo 427  
  caricamento di variabili in 427  
  compilazione con testo esterno 430  
  confronto tra nomi di istanze e di variabili 421  
  controllo di contenuti multimediali incorporati 488  
  creazione dinamica in fase di runtime 420, 422  
  definizione 876  
  dinamico 417  
  e testo HTML 469  
  evitare conflitti tra nomi di variabili 422  
  formattazione 458  
  formattazione con CSS 461  
  formattazione in HTML 420  
  gestione 424  
  impostazione dello spessore 441  
incorporamento di clip filmato in 487  
incorporamento di file SWF o di immagine 486  
incorporamento di immagini selezionabili in 490  
informazioni 417  
modifica della posizione 424  
modifica delle dimensioni 425  
nomi di istanze 421  
proprietà predefinite 460  
specifiche delle dimensioni di immagine 488  
visualizzazione di proprietà per l'esecuzione del  
  debug 788  
*Vedere anche* classe TextField, classe TextFormat e  
  classe TextField.StyleSheet  
campo con distanza a campionamento adattivo (ADF)  
  446  
carattere barra rovesciata, nelle stringhe 503  
carattere di avanzamento pagina 503  
carattere di avanzamento riga 503  
carattere escape 503  
carattere Tab 503  
carattere virgoletta doppia, nelle stringhe 502, 503  
carattere virgoletta semplice, nelle stringhe 502, 503  
caratteri  
  aggiunta e rimozione 433  
  condivisione 442  
  definizione 433, 870  
  incorporati, aggiunta e rimozione 433  
  informazioni 433  
  valori di taglio 449  
caratteri dispositivo  
  definizione 445, 871  
  effetto maschera 410  
caratteri incorporati  
  aggiunta e rimozione 433  
  incorporamento di un simbolo di carattere 436  
  uso con campi di testo 434  
  uso con classe TextField 441  
caratteri speciali 343  
caricamento  
  contenuti multimediali esterni 646  
  visualizzazione di file XML 432  
caricamento dati  
  da server 372  
  variabili 373  
Cascading Style Sheets. *Vedere* CSS  
casella degli strumenti Azioni, voci in giallo nella 55  
chiamata di metodi 339  
cicli  
  creazione e uscita 122  
  do..while 128

for..in 125  
nidificati 128  
uso 119  
while 126  
classe BitmapData  
  applicazione di filtri a 551  
Classe Delegate  
  informazioni 329  
  uso 330  
Classe di andamento Rimbalzo 534  
classe String  
  charAt(), metodo 505  
  concat(), metodo 509  
  e metodi substr() e substring() 511  
  informazioni 492, 500  
  length, proprietà 504, 507  
  split(), metodo 509  
  toLowerCase() e toUpperCase(), metodi 507  
  toString(), metodo 507  
Classe TextField  
  creazione di testo scorrevole 491  
  uso 418  
Classe TextField.StyleSheet 461  
  e proprietà TextField.styleSheet 461, 466  
classe TextField.StyleSheet  
  creazioni di stili di testo 466  
  CSS 463  
classe TextFormat  
  informazioni 452  
  uso 458  
classe TransitionManager  
  e andamento 527  
  informazioni 527  
  uso 531  
classe XML, metodi 711  
classi  
  accesso alle proprietà incorporate 274  
  assegnazione ai clip filmato 412  
  assegnazione di nomi alle classi 240  
  assegnazione di un'area di validità 811  
  assegnazione di un'istanza in Flash 255  
  chiamata dei metodi di un oggetto incorporato 275  
  classi di flash.display 269  
  classi di flash.external 269  
  classi di flash.filters 270  
  classi di flash.geom 271  
  classi di flash.net 271  
  classi di flash.text 272  
  classi di mx.lang 272  
  classi di System e TextField 272  
come progetti 201  
come tipi di dati 198  
compilazione ed esportazione 257  
confronto con interfacce 294  
confronto con pacchetti 201  
controllo dell'accesso dei membri 248  
creazione di classi dinamiche 233  
creazione di istanze 198  
creazione di un file classe 217  
creazione di un'istanza di 254  
creazione di una nuova istanza di una classe  
  incorporata 274  
creazione di una sottoclasse 281  
creazione e inserimento in pacchetti 240  
definizione 273  
di primo livello 265  
documentazione 250  
e area di validità 237, 260  
e file ASO 258  
e funzioni di costruzione 243  
e polimorfismo 287  
e variabili di istanza. 248  
ed ereditarietà 279  
ereditarietà, esempio 282  
esclusione di classi incorporate 276  
importazione 212  
importazione ed esportazione 252  
incapsulamento 236  
incorporate e di primo livello 263  
incorporate, informazioni su 199  
inizializzazione delle proprietà in fase di runtime 413  
membri delle classi 224  
membri statici di classi incorporate 275  
metodi e proprietà 218  
metodi e proprietà pubblici, privati e statici 220  
metodi e proprietà statici 222  
metodi getter/setter 229  
organizzazione in pacchetti 200  
percorsi di classe 214  
precaricamento 277  
procedure ottimali per la scrittura 239  
proprietà delle 220  
proprietà e metodi privati 222  
riferimenti risolti dal compilatore 217  
risoluzione dei riferimenti alle classi 217  
scrittura di classi personalizzate 208  
scrittura di metodi e proprietà 245  
scrittura di un esempio personalizzato 237  
sostituzione di metodi e proprietà 285

superclasse 281  
uso delle classi incorporate 273  
utilizzo dei metodi getter/setter 230  
utilizzo di classi personalizzate 211  
utilizzo di classi personalizzate in Flash 252  
vantaggi dell'utilizzo 199  
*Vedere anche* classi, incorporate  
classi dinamiche 233  
clip filmato  
  aggiunta di parametri 396  
  applicazione del filtro bagliore 545  
  assegnazione di stati del pulsante a 324  
  assegnazione di una classe personalizzata a 255  
  associazione a simboli sullo stage 394  
  associazione ai gestori on() e onClipEvent() 319  
  attivazione tramite tastiera 622  
  avvio e interruzione 616  
  caricamento di file MP3 in 652  
  caricamento di file SWF e JPEG 647  
  condivisione 394  
  confronto tra metodi e funzioni 382  
  controllo 382  
  creazione di sottoclassi 412  
  creazione di un'istanza vuota 392  
  creazione in fase di runtime 391  
  determinazione della profondità dei 400  
  determinazione della successiva profondità disponibile 399  
  dissolvenza con codice 516  
  duplicazione 393  
  e istruzione with 384  
  \_root, proprietà 386  
  elenco degli oggetti 787  
  elenco delle variabili 787  
  eliminazione 393  
  filters, proprietà 571  
  funzioni 383  
  gestione della profondità 398  
  identificazione di create dinamicamente 84  
  incorporamento in campi di testo 486  
  inizializzazione delle proprietà in fase di runtime 413  
  invocazione di metodi 383  
  metodi, elenco 383  
  metodi, uso per il disegno di forme 590  
  modifica delle proprietà durante la riproduzione 388  
  modifica delle proprietà nel Debugger 779  
  modifica di colore e luminosità 518  
  nidificato, definizione 381  
nome di istanza, definizione 381  
principale, definizione 381  
proprietà 388  
  proprietà, inizializzazione in fase di runtime 413  
regolazione del colore 624  
richiamo di più metodi 384  
rilevamento di collisioni 628  
rimozione 393  
ripetizione ciclica di elementi secondari 121  
secondario, definizione 381  
sfondo 408  
tipi di dati 339  
trascinamento 390  
uso come maschere 409  
*Vedere anche* File SWF  
clip filmato nidificati, definizione 381  
clip filmato principali 381  
clone(), metodo  
  informazioni 583  
  uso 583  
code  
  esempi, copiare e incollare 14  
  scorrimento delle righe 783  
  selezione di una riga 781  
codice  
  formattazione 54, 55  
  ritorno a capo automatico 56  
  visualizzazione dei numeri di riga 56  
codice di un oggetto, definizione 874  
codici tasto, ASCII  
  accesso 621  
  altri tasti 848  
  tasti di lettere e numeri 844  
  tasti funzione 847  
  tastierino numerico 846  
codifica del testo 61  
codifica di caratteri 493  
collegamento, clip filmato 394  
collisioni, rilevamento 628  
  tra clip filmato 629  
  tra clip filmato e un punto sullo stage 629  
colori  
  nella casella degli strumenti Azioni 55  
  valori, impostazione 624  
Comando Elenca variabili 787  
commenti  
  all'interno di classi 99  
  disordinati o raggruppati 96  
  e colorazione della sintassi 96  
  finali 98

informazioni 96  
multiline 97  
nei file di classe 250  
procedure ottimali 804  
riga singola 97  
scrittura nei file di classe 806

Componente FLVPlayback  
creazione di cue point con cui lavorare 671  
e cue point 670  
e seek(), metodo 674  
ricerca del cue point 675, 676  
ricerca di una durata specifica 674  
utilizzo di cue point con 671

componenti di testo 417

componenti, convenzioni di codifica 803

comportamenti  
informazioni 66  
transizione Zoom 529

comportamento transizione Zoom 529

comunicazione con Flash Player 719

concatenamento  
convenzioni di codifica 803  
identificatore 394, 412

concatenazione di stringhe 342

condivisione dei caratteri  
informazioni 442

condizioni  
scrittura 108

condizioni, informazioni su 108

contatori, ripetizione di azioni 120, 121

contentore di FlashPlayer  
definizione 872

contenuti multimediali esterni 645  
caricamento di file di immagine e di file SWF 647  
caricamento di file MP3 684  
caricamento di file SWF e file di immagine esterni 682  
caricamento di file SWF e JPEG 647

creazione di animazioni per barre di avanzamento 681  
e linea temporale principale 651

file MP3 652  
informazioni sul caricamento 646  
motivi per l'uso 645  
precaricamento 665, 681  
ProgressBar, componente 650  
riproduzione di file FLV 660

controlli della tastiera  
per attivare i clip filmato 622  
Prova filmato 772

controllo  
dei dati caricati 691  
sintassi e punteggiatura 59

convenzioni di assegnazioni di nomi 793  
booleano 799  
classi e oggetti 800  
funzioni e metodi 800  
interfacce 803  
pacchetti 200, 802  
variabili 52, 796

convenzioni di codifica  
ActionScript 807  
componenti 803

convenzioni tipografiche 13

convenzioni, assegnazioni di nomi 793

conversione di tipi di dati 334

costanti  
definizione 870  
informazioni 100  
procedure ottimali 799  
uso 101

creazione di istanze  
definizione 198  
di oggetti 274

creazione di oggetti 274

creazione di script di ActionScript  
super (prefisso) 818  
trace 817  
with (istruzione) 820

creazione di stringhe 501

criteri, file  
definizione 762  
devono essere denominati crossdomain.xml 762  
*Vedere anche sicurezza*

CSM  
informazioni 446  
informazioni sui parametri 446

CSS  
applicazione a campi di testo 466  
applicazione di classi di stile 468  
assegnazione di stili a tag HTML incorporati 469  
caricamento 464  
combinazione di stili 468  
definizione di stili in ActionScript 466  
e classe TextField.StyleSheet 463  
esempio con tag HTML 470  
esempio con tag XML 473  
formattazione del testo con 461  
proprietà supportate 462  
uso per la definizione di nuovi tag 472

cue point  
creazione 671  
navigazione, evento e ActionScript 667  
operazioni 670  
tracciamento 668  
uso 667  
visualizzazione 670  
cursori, creazione personalizzati 618  
CustomFormatter, classe  
informazioni 638  
uso 639

**D**

dati  
associazione con componenti 632  
barra di caricamento e avanzamento 692  
definizione 333  
e variabili 333  
informazioni 333  
organizzazione in oggetti 375  
dati caricati, controllo 691  
dati, esterni 689, 733  
accesso tra file SWF di domini diversi 760, 764  
controllo del caricamento 691  
e messaggi 719  
e oggetto LoadVars 697  
e oggetto XMLSocket 717  
e script sul lato server 695  
e XML 709  
invio e caricamento 690  
sicurezza, funzioni 753  
Debug Player 771  
Debugger  
attivazione del debug remoto 775  
Elenco di controllo 777  
Flash Debug Player 771  
impostazione dei punti di interruzione 780  
pulsanti nel 783  
Scheda Proprietà 779  
selezione dal menu di scelta rapida 775  
uso 771  
variabili 776  
descrizione comandi. *Vedere* suggerimenti sul codice  
determinazione della posizione del puntatore del mouse  
619  
disegno  
mediante codice 590  
distinzione tra maiuscole e minuscole

e Flash Player versione 79  
informazioni 78  
do..while 128  
Documentazione PDF, dove trovare 16  
documentazione, altro materiale di riferimento 18  
DOM (Document Object Model), XML 709  
drawingAPI  
mediante classi Tween e TransitionManager 604  
duplicazione, clip filmato 393

**E**

editor del metodo di input  
informazioni 497  
uso 498  
Editor di ActionScript 869  
effetti  
brillantezza 574  
dissolvenza 516  
disturbo 585  
interpolazione luminosità 521  
luminosità e colore 518  
metodi di fusione 588  
panoramica di un'immagine 524  
scala di grigi 520  
effetti. *Vedere* filtri  
effetto di disturbo 585  
elementi, di array 130  
Elenca oggetti, comando 787  
endpoint 636  
ereditarietà  
e OOP 205  
e sottoclassi 280  
esempio 282  
informazioni 279  
errore legato a esaurimento di memoria 552  
esecuzione del debug 771  
con l'istruzione trace 789  
da una postazione remota 774  
Debug Player 771  
elenco degli oggetti 787  
elenco delle variabili 787  
messaggi di errore del compilatore 835  
proprietà del campo di testo 788  
uso del pannello Output 785  
esportazione di script e codifica del linguaggio 61  
espressioni  
definizione 872  
gestione dei valori 143

- espressioni condizionali 118
- eventi
- definizione 311, 871
  - e clip filmato 411
  - trasmissione 324
- evento utente 311
- extends, parola chiave 280
- informazioni 280
  - sintassi 281
- Extensible Markup Language. *Vedere XML*
- ExternalInterface, classe
- informazioni 723
  - uso 724
- F**
- fase di compilazione, definizione 14
- file classe esterni
- uso di percorsi di classe per individuare 214
- file di classe
- linee guida per l'organizzazione 814
  - strutturazione 813
- file di configurazione 70
- file di esempio, informazioni su 15
- file Flash 4, apertura con Flash 8 853
- File FLV
- Vedere anchevideo*
- file MP3
- caricamento 652, 654
  - caricamento in clip filmato 652
  - creazione di una barra di avanzamento 684
  - lettura dei tag ID3 656
  - precaricamento 655, 665
  - tag ID3 656
- File SWF
- mantenimento, dimensioni originali 720
  - ridimensionamento a, Flash Player 720
- file SWF caricati
- identificazione 85
  - rimozione 385
- file XLIFF 495
- File XML, aggiornamento per l'installazione di Flash 8 10
- file, caricamento 700
- FileReference, classe
- creazione di un'applicazione 703
  - e download (), metodo 701
  - e sicurezza 702
  - informazioni 700
- filtri
- animazione 582
  - applicazione a istanze 551
  - array 580
  - definizione 545
  - disturbo 585
  - e ActionScript 553
  - e prestazioni 552
  - e trasparenza 554
  - e uso della memoria 552
  - ed errore legato a esaurimento di memoria 552
  - ed errori di gestione 552
  - filtro bagliore 545
  - gestione mediante codice 579
  - modifica del livello di luminosità 574
  - modifica delle proprietà 549
  - nozioni fondamentali sui pacchetti 547
  - ottenimento e impostazione 549
  - regolazione delle proprietà 580
  - rotazione e inclinazione 550
  - rotazione, inclinazione e modifica in scala 551
- filtro bagliore
- come animare 545
  - informazioni 562
  - uso 562
- filtro bagliore con gradiente
- informazioni 563
  - uso 564
- filtro di convoluzione
- informazioni 576
  - informazioni sull'applicazione 576
  - uso 576
- filtro mappa di spostamento
- applicazione a un'immagine 586
  - informazioni 577
  - mediante classe BitmapData 586
  - uso 578
- filtro matrice colore
- informazioni 573
  - uso 520, 574
- filtro sfocatura
- animato mediante la classe Tween 582
  - informazioni 555
  - uso e animazione 556
- filtro smussatura
- informazioni 565
  - uso 566
- filtro smussatura con gradiente
- applicazione 572
  - applicazione a un clip filmato: 572

array dei colori 568  
array ratio 569  
distribuzione del colore 568  
e proprietà blurX e blurY 568  
e proprietà knockout e type 568  
e proprietà strength 568  
e riempimento 567  
e riempimento del clip filmato 571  
ed evidenziazione 570  
informazioni 567  
uso 570  
valore ratio e angle 570  
finestra di messaggio, visualizzazione 720  
finestra Script  
definizione 875  
informazioni 36, 38  
informazioni sul file XML dei punti di interruzione 782  
opzioni di menu 43  
scrittura di codice 40  
Flash 8, funzioni di ActionScript nuove e modificate 19  
Flash Player  
acquisizione della versione più recente 790  
classi, informazioni 264  
comunicazione 719  
e ActionScript 824  
impostazioni di pubblicazione 74  
menu di tipo normale, visualizzazione 720  
metodi 722  
ridimensionamento, file SWF 720  
standard di codifica 824  
versione per il debug 772  
visualizzazione o disattivazione, menu di scelta rapida 720  
visualizzazione, schermo intero 720  
Flash Player 4  
creazione di contenuto per 852  
Flash Player 7  
nuovo modello di sicurezza 754, 761, 768  
porting di script esistenti 734  
Flash Player 8  
elementi del linguaggio ActionScript nuovi e modificati 22  
elementi di linguaggio sconsigliati 27  
funzioni dell'editor di ActionScript nuove e modificate 28  
Flash Player precedenti, destinazione 851  
FlashType  
informazioni 443  
supporto Flash Player 443  
FlashVars  
informazioni 427  
uso per visualizzazione di testo 428  
FLV, file  
caricamento di file esterni in fase di runtime 662  
configurazione del server per FLV 679  
creazione di un banner FLV 663  
creazione di una barra di avanzamento 686  
cue point 667, 668  
e Macintosh 680  
lavoro con i cue point 670  
metadati 677  
navigazione con il codice 674  
precaricamento 665  
precaricamento di video esterni 665  
video esterni 660  
fogli di stile *Vedere* CSS  
for 123  
esempio 137  
for..in 125  
formattazione del testo  
definizione 876  
informazioni 452  
uso 453  
formattazione di codice 54, 55  
formatter personalizzati  
informazioni 637  
uso 637  
frequenza fotogrammi  
e onEnterFrame 515  
informazioni 515  
mediante la classe Tween 539  
scelta 515  
fscommand(), funzione  
comando e argomenti 720  
comunicazione con Director 722  
uso 719  
function  
blocco di funzione 175  
funzione anonima  
definizione 869  
scrittura 176  
uso 179  
funzione letterale  
definizione 872  
informazioni 179  
ridefinizione 179  
funzioni  
asincrone 691

assegnazione di un nome 183  
blocco di funzione 176  
chiamata di funzioni di primo livello 174  
come scatola nera 170  
confronto con metodi 194  
confronto tra funzioni con nome e anonime 185  
conversione 334  
creazione e chiamata 184  
definizione 180, 872  
definizione di funzioni globali e associate alla linea temporale 180  
di callback 177  
di costruzione 179  
di primo livello 172  
esempio 874  
formato standard delle funzioni con nome 175  
funzione di costruzione 856  
funzione letterale 179  
Identificazione e chiamata delle funzioni definite dall'utente 181  
in un file di classe 186  
incorporate e di primo livello 173  
informazioni 169  
nidificati 192  
passaggio di parametri 188  
per il controllo di clip filmato 383  
personalizzate 169  
procedure ottimali 821  
restituzione di valori da 190  
riutilizzo 183  
scrittura di funzioni anonime 176  
scrittura di funzioni con nome 175  
sintassi di funzioni con nome 170  
tipi di 171  
uso in Flash 183  
utilizzo delle funzioni con nome 175  
utilizzo di variabili in 187  
funzioni con nome 176  
definizione 874  
funzioni definite dall'utente  
definizione 876  
scrittura 181  
funzioni di callback  
definizione 870  
scrittura 177  
funzioni di conversione e tipi di dati 335  
funzioni di costruzione  
definizione 870  
esempio 856  
scrittura 179

funzioni di primo livello  
definizione 876  
funzioni personalizzate 169

## G

garbage collection 816  
gestione degli errori e filtri 552  
gestione dei numeri 341  
gestore di eventi, metodi  
controllo dei dati XML 691  
gestori on() e onClipEvent() 319  
area di validità 326  
associazione ai clip filmato 319  
gestori. *Vedere* gestori di eventi  
getAscii(), metodo 621

## H

hitTest(), metodo 628  
HTML  
assegnazione dello stile a tag incorporati 469  
campo di testo 420  
esempio di utilizzo degli stili 470  
tag compresi tra virgolette 476  
tag supportati 477  
uso del tag <img> per applicare un flusso al testo 475, 479, 486  
uso di CSS per la definizione di tag 472  
uso in campi di testo 476  
HTTP, protocollo  
comunicazione con script sul lato server 695  
con metodi ActionScript 690  
HTTPS, protocollo 690

## I

icone  
nel Debugger 783  
sopra il riquadro dello script 41  
IDE (ambiente di sviluppo integrato), definizione 872  
identificatori, definizione 872  
IME (input method editor)  
informazioni 497  
uso 498  
immagine in scala di grigi 520  
immagine JPEG a scaricamento progressivo,  
definizione 874  
immagini

applicazione dei metodi di fusione 588  
 caricamento in clip filmato 388  
 incorporamento in campi di testo 486  
 Vedere anche contenuti multimediali esterni  
 immagini vettoriali 876  
 import  
    classi multiple nel pacchetto 547  
    informazioni sull'istruzione 547  
    uso del carattere jolly 548  
 importazione  
    file di classe 212  
    script e codifica del linguaggio 61  
 impostazioni di pubblicazione  
    ActionScript 67  
    modifica 67  
    modifica del percorso di classe 68  
    scelta della versione di Flash Player 74  
 encapsulamento  
    informazioni 207  
    uso 236  
 incorporamento caratteri, finestra di dialogo  
    uso 438  
 incorporate, funzioni 173  
 indirizzi IP  
    file di criteri 763  
    sicurezza 753  
 informazioni, passaggio tra file SWF 690  
 inizializzazione delle proprietà del clip filmato 413  
 inizializzazione, creazione di script di ActionScript 816  
 inserimento di oggetti 378  
 interattività, file SWF  
    creazione 613  
    tecniche di 618  
 interfacce  
    assegnazione di un nome 296  
    creazione 297  
    creazione come tipo di dati 299  
    definizione ed implementazione 296  
    e OOP 206  
    esempio 303  
    informazioni 293  
    interfaccia complessa, esempio 305  
    nozioni fondamentali sull'ereditarietà e 301  
 interface, parola chiave 295  
 interpolazioni  
    aggiunta mediante ActionScript 531  
    aggiunta mediante comportamenti 528  
 Interruzione di clip filmato 616  
 invio di informazioni  
    a file remoti 690  
 formato con codifica URL 690  
 in formato XML 690  
 tramite TCP/IP 690  
 istanze 516  
 applicazione di filtri a 551  
 definizione 273, 872  
 e OOP 205  
 identificazione di istanze nidificate 83  
 identificazione dinamica 84  
 target 82  
 istruzione try..catch..finally, scrittura 115, 833  
 istruzioni  
    composti 107, 830  
    condizionali 109, 828  
    definizione 76, 875  
    for 831  
    if 109  
    if..else 110  
    if..else if 111  
    importazione 203  
    informazioni 106  
    istruzioni trace 789  
    linee guida per la scrittura 106  
    switch 113  
    try..catch..finally 115, 833  
    while e do while 831  
    with 820  
 istruzioni cicliche 120, 121  
 istruzioni composte 107  
    scrittura 830  
 istruzioni condizionali  
    scrittura 828  
 istruzioni for, scrittura 831  
 istruzioni if..else if, scrittura 111  
 istruzioni if..else, scrittura 110  
 istruzioni switch  
    convenzioni 832  
    uso 113  
 istruzioni trace, creazione di script di ActionScript 817

## J

JavaScript  
    alert, istruzione 789  
    e ActionScript 77  
    e Netscape 722  
    invio di messaggi 720  
    standard internazionale 77  
 JPEG, file

- caricamento in clip filmato 388, 647  
incorporamento in campi di testo 486
- ## L
- layout del testo 452  
linee 598  
lingue, uso di più lingue negli script 61  
listener (sintassi) 833  
listener di eventi 314  
    area di validità 325  
    classi che possono trasmettere 315  
LiveDocs, informazioni su 16  
livelli  
    caricamento 385  
livelli, identificazione della profondità 85  
loadMovie(), funzione 691  
loadVariables(), funzione 691  
LoadVars, classe  
    caricamento di variabili da file di testo 431  
    uso 696  
    uso per visualizzazione di testo 430  
    verifica HTPP, stato 699  
Locale, classe  
    informazioni 495  
    uso 495  
loops  
    for 123
- ## M
- Macromedia Director, comunicazione 722  
maschere 409  
    e applicazione dell'effetto maschera ai canali alfa  
        411  
    e caratteri dispositivo 410  
    script da creare 605  
    tratti ignorati 409, 591  
materiale di riferimento aggiuntivo 15  
MediaPlayback, componente  
    utilizzo di cue point con 673  
membri (metodi e proprietà)  
    pubblici, privati e statici 220  
membri delle classi 205, 275  
    informazioni 205  
membri statici 275  
membri statici. *Vedere* membri delle classi  
memorizzazione delle bitmap nella cache  
    attivazione 402
- definizione 402  
e applicazione dell'effetto maschera ai canali alfa  
    411  
e filtri 549  
informazioni 401, 526  
memorizzazione di un clip filmato nella cache 406  
opaqueBackground, proprietà 402  
quando evitarlo 405  
quando utilizzarla 404  
scrollRect 402  
superfici 401  
vantaggi e svantaggi 403  
menu a comparsa Opzioni di visualizzazione 56, 57  
messina pausa (scorrimento) del codice 783  
messaggi di errore 835  
metadati  
    informazioni 677  
    uso 677  
metodi  
    asincroni 691  
    assegnazione di un nome 195  
    confronto con funzioni 194  
    definizione 192, 873  
    degli oggetti, richiamo 275  
    e array 192  
    informazioni 169, 192  
    per il controllo di clip filmato 383  
    private 222  
    pubblici 221  
    static 222  
    tipi di 171  
metodi del gestore di eventi  
    area di validità 325  
    assegnazione di funzioni a 314  
    associazione a pulsanti o clip filmato 319  
    associazione agli oggetti 322  
    definiti dalle classi ActionScript 312  
    definizione 311, 871  
    e on() e onClipEvent() 319  
    in ActionScript 2.0 328
- metodi di disegno  
    *Vedere anche* API di disegno
- metodi di fusione  
    applicazione 588  
    informazioni 587
- metodi di fusione. *Vedere* metodi di fusione
- metodi getter  
    informazioni 229  
    uso 230
- metodi setter

- informazioni 229  
uso 230
- metodi TextField, uso 441
- metodo getURL() 617
- MIME, formato standard 696
- modalità Assistente script  
definizione 875  
informazioni 63
- modelli di progettazione  
incapsulamento 236  
Singleton 226
- modello a eventi  
per gestori on() e onClipEvent() 319  
per i metodi del gestore di eventi 312  
per listener di eventi 315
- Modello di progettazione Singleton 226
- modifica di ActionScript  
a capo automatico 56  
bloccare gli script 63  
controllo sintassi 59  
evidenziazione sintassi 55  
importazione ed esportazione degli script 60  
numeri di riga 56  
strumento Trova 59  
suggerimenti sul codice 52  
tasti di scelta rapida Esc 57  
visualizzazione dei caratteri nascosti 58
- modifica in scala a 9 porzioni  
attivazione 608  
informazioni 606  
nozioni fondamentali 606  
scale9Grid, proprietà 608  
uso 609
- modulazione continua del tratto 446
- MovieClip, classe  
e proprietà scale9Grid 608  
filters, proprietà 549  
metodi di disegno 590  
proprietà blendMode 587  
regolazione della proprietà filters 580
- N
- navigazione  
controllo 613  
passaggio a un fotogramma o a una scena 615
- Netscape, metodi JavaScript supportati 722
- NetStream, classe  
e gestore onMetaData 677
- utilizzo del gestore onMetaData 677
- nodi 709
- nodo glyphRange, informazioni 439
- nome completo  
definizione 547  
uso 547
- nomefilmato\_DoFSCommand, funzione 720
- nomi di domini e sicurezza 753
- nomi di istanze  
confronto con nomi di variabili 421  
definizione 381, 873  
e percorsi target 81
- numeri di riga nel codice, visualizzazione 56
- numeri, gestione con i metodi 341
- O
- obsoleti, operatori di Flash 4 841
- oggetti  
accesso alle proprietà 274  
creazione 274, 354, 375  
creazione in Flash 376  
definizione 874  
dissolvenza in uscita 516  
organizzazione dei dati in array 377  
richiamo di metodi 275  
ripetizione ciclica di elementi secondari 121  
standard di codifica 808  
tipi di dati 341
- oggetti listener 314  
annullamento della registrazione 316
- oggetto broadcaster 314
- oggetto LoadVars, creazione 697
- ombra esterna, filtro  
animazione 560  
applicazione a immagini trasparenti 561  
e metodo clone() 583  
informazioni 557  
uso 558
- onEnterFrame e frequenza fotogrammi 515
- OOP  
e encapsulamento 207  
e interfacce 206  
e oggetti 204  
e polimorfismo 207  
ed ereditarietà 205  
informazioni 198, 204  
Istanze e membri di classe 205  
progettazione 236

scrittura di classi personalizzate 208  
opaqueBackground, proprietà  
definizione 402  
uso 409  
operandi 143  
operatore condizionale 118  
operatori  
additivi 155  
assegnazione 145, 161  
associatività 146  
combinazione con i valori 143  
condizionali 118, 158, 167  
confronto 150  
definizione 874  
di spostamento bit a bit 164  
espressioni matematiche 143  
forma suffissa 153  
gestione dei valori 145  
informazioni 143  
logici 162, 163  
logici bit a bit 165  
moltiplicativi 155  
numerici 155  
obsoleti 841  
operandi 143  
precedenza e associatività 146  
punto e operatore di accesso agli array 151  
relazionali 157  
relazionali e di uguaglianza 158  
uguaglianza 157, 158  
unari 154  
uso in Flash 167  
utilizzo con le stringhe 149  
utilizzo dell'assegnazione 162  
operatori di accesso agli array, verifica delle coppie corrispondenti 60  
operatori di confronto 158  
operatori di uguaglianza 158  
operatori relazionali 158  
opzione Blocca lo script nel pannello Azioni 64  
opzioni colori sintassi, impostazione nel pannello Azioni 56  
opzioni di rendering di testo 445  
ordine delle operazioni 590  
ordine di esecuzione (operatore)  
operatori, associatività 146  
priorità degli operatori 146  
organizzazione degli script  
ActionScript 1.0 e ActionScript 2.0 73  
associazione agli oggetti 808  
convenzioni di codifica 807  
origini esterne, connessione con Flash 689, 733  
ottenere informazioni da file remoti 690

## P

pacchetti  
assegnazione di un nome 200  
confronto con classi 201  
definizione 874  
importazione 203  
informazioni 200  
operazioni 202, 547  
Pannello Output 785  
Comando Elenca variabili 787  
copia del contenuto 786  
e istruzione trace 789  
Elenca oggetti, comando 787  
opzioni 785  
visualizzazione 785  
pannello Stringhe 494  
parametri 175, 874  
parentesi graffe, verifica delle coppie corrispondenti 60  
parentesi, verifica delle coppie corrispondenti 60  
parole chiave  
\_root 87  
definizione 873  
elenchi 104  
extends 280  
informazioni 100  
interface 295  
this 86  
uso 103  
parole riservate  
altre raccomandazioni 106  
elenchi 104  
informazioni 104  
nomi di classi incorporate 105  
parole riservate in futuro 105  
parole riservate.  
*Vedere anche* parole chiave  
passaggio a un URL 617  
password e debug remoto 774  
percorsi relativi 86  
percorso di classe  
definizione 214  
eliminare una directory da 215  
globale 215  
informazioni 68, 74

livello di documento 216  
modifica 68  
ordine di ricerca di 217  
percorso target  
definizione 876  
e indirizzamento dell'istanza 82  
e istanze nidificate 83  
e sintassi del punto 81  
inserimento 65, 87  
uso 181  
utilizzo dei pulsanti 87  
più lingue, uso negli script 61  
polimorfismo  
informazioni 207  
uso 287  
posizione del mouse, determinazione 619  
precedenza e associatività degli operatori 146  
preferenza di codifica predefinita 61  
prefissi, super 818  
prestazioni  
e filtri 552  
e frequenza fotogrammi 515  
memorizzazione delle bitmap nella cache 526  
procedure ottimali  
ActionScript 1 e ActionScript 2,0 73  
area di validità 809  
assegnazione di nomi a classi e oggetti 800  
assegnazione di nomi a funzioni e metodi 800  
assegnazione di nomi ai pacchetti 802  
assegnazione di nomi alle costanti 799  
assegnazione di nomi alle variabili booleane 799  
assegnazione di nomi di interfacce 803  
assegnazione di un nome alle variabili 796  
commenti 804  
commenti nelle classi 806  
convenzioni di codifica 793  
funzioni 821  
profili di carattere 449  
profondità  
definizione 398  
determinazione dell'istanza nella 399  
determinazione della successiva disponibile 399  
determinazione per i clip filmato 400  
gestione 398  
programmazione orientata agli oggetti 204  
Programmazione orientata agli oggetti. *Vedere OOP*  
proiettori, esecuzione applicazioni da 720  
proprietà  
definizione 874  
degli oggetti, accesso 274  
dei clip filmato 388  
inizializzazione in fase di runtime 413  
private 222  
public 221  
static 222  
proprietà degli oggetti  
assegnazione di valori 274  
Proprietà di concatenamento, finestra di dialogo 394, 412  
Proprietà FlashVars  
uso 370  
proprietà FlashVars  
informazioni 427  
Prova filmato  
scelte rapide da tastiera 772  
Unicode 772  
prova. *Vedere* esecuzione del debug  
puntatore del mouse. *Vedere* cursori  
puntatore. *Vedere* cursori  
punti di interruzione  
e file esterni 780  
impostazione nel Debugger 780  
informazioni 780  
XML, file 782  
punti di interruzione, impostazione e rimozione  
in pannello Azioni 780  
nella finestra Script 780  
punto di registrazione, e immagini caricate 388

## R

remoto  
esecuzione del debug 774  
file, comunicazione 690  
siti, connessione continua 717  
rendering di caratteri  
informazioni 443  
metodi 444  
opzioni 445  
return (istruzione) 831  
rientro del codice, attivazione 55  
rilevamento di collisioni 628  
rilevamento di tasti premuti 621  
rimozione  
clip filmato 393  
file SWF caricati 385  
ripetizione di azioni, utilizzando i cicli 119  
riproduzione di clip filmato 616  
Riquadro dello script

pulsanti sopra 41  
riquadro dello script  
definizione 875  
risoluzione dei problemi *Vedere* esecuzione del debug  
risorse in linea 18  
ritorno a capo automatico nel codice, attivazione 56  
proprietà \_root e clip filmato caricati 386  
runtime, definizione 14

**S**

sblocco degli script nel pannello Azioni 65  
scala 9  
informazioni 606  
scala 9. *Vederemodifica* in scala a 9 porzioni  
Scheda Controllo, Debugger 777  
Scheda Proprietà, Debugger 779  
scheda Variabili, Debugger 776  
scorrevole  
e memorizzazione delle bitmap nella cache 526  
testo 491  
scorrimento delle righe di codice 783  
script  
blocco 64  
dove scrivere 31  
esecuzione del debug 771  
eventi associati ai clip filmato 33  
eventi di tastiera 33  
importazione ed esportazione 61  
informazioni sugli eventi 32  
organizzazione del codice 34  
porting in Flash Player 7 734  
prova 771  
risoluzione dei problemi relativi alla visualizzazione  
del testo 61  
script di fotogramma 33  
scrittura per la gestione di eventi 35  
tasti di scelta rapida per gli script bloccati 64  
script di fotogramma  
informazioni 33  
script di fotogrammi  
definizione 872  
Script navigator 37  
script sul lato server  
creazione 707  
linguaggi 690  
XML, formato 711  
scrittura di sintassi e istruzioni  
listener 833

return 831  
switch 832  
scrollRect, proprietà 402  
secondario  
clip filmato, definizione 381  
node 709  
security  
Compatibilità di Flash Player 734  
sequenze di caratteri *Vedere* stringhe  
sequenze di escape 343  
Server Web IIS 6.0 679  
server, apertura di una connessione continua 717  
set di caratteri  
creazione di set personalizzati 439  
set di caratteri personalizzati, creazione 438, 439  
setInterval  
e frequenza fotogrammi 515  
uso 517  
setRGB, metodo 624  
sicurezza  
accesso a dati tra domini diversi 760  
e file di criteri 762  
e porting degli script in Flash Player 7 754, 761, 768  
loadPolicyFile 764, 765  
tra domini diversi 753  
simboli di carattere, incorporamento 436  
sintassi  
barra 87  
controllo 59  
distinzione tra maiuscole e minuscole 79, 80  
sintassi che segue i due punti, definizione 346  
sintassi del punto (notazione del punto) 81  
sistema  
evento, definizione 311  
requisiti, per ActionScript 2.0 10  
socket, connessioni  
informazioni 717  
script di esempio 718  
sottoclassi  
creazione di script 281  
creazione per i clip filmato 412  
esempio 282  
sottoclassi, informazioni 280  
Specifica ECMA-262 77  
Stage, associazione di simboli a 394  
standard di codifica UTF-16 493  
stili  
linea 598  
tratti ed estremità 598

stili delle estremità, impostazione 598  
stili di estremità  
impostazione 598  
informazioni 598  
stili di linea  
alfa 601  
capsStyle e jointStyle 602  
color 600  
e API di disegno 598  
informazioni 598  
miterLimit 604  
modifica in scala 601  
parametri 599  
pixelHinting 601  
stili di tratti ed estremità 598  
thickness 599  
stili di tratti 598  
stili, tratti ed estremità 598  
stringhe 342  
analisi 504  
confronto 504  
confronto con altri tipi di dati 506  
conversione di maiuscole e minuscole 508  
conversione e concatenazione 507  
creazione 501  
creazione di un array di sottostringhe 509  
definizione 493, 875  
determinazione della lunghezza 504  
esame di caratteri 505  
impostazione del confronto di tipo 507  
informazioni 492  
restituzione di sottostringhe 511  
ricerca della posizione dei caratteri 512  
ricerca di una sottostringa 511  
spostamento in 505  
uso 502  
suggerimenti sul codice 48  
attivazione 47, 51, 53  
informazioni 47  
specifiche delle impostazioni per 48  
uso 48  
visualizzazione manuale 51  
suoni  
*Vedere anche* contenuti multimediali esterni  
super (prefisso) 818  
superclasse 281  
superficie  
caching bitmap 875  
definizione 401  
SWF, file  
caricamento e scaricamento 385  
caricamento in clip filmato 647  
controllo in Flash Player 722  
creazione di controlli audio 625  
incorporamento in campi di testo 486  
inserimento su pagina Web 617  
passaggio a un fotogramma o a una scena 615  
passaggio di informazioni tra i file 690  
*Vedere anche* clip filmato

## T

Tab e Prova filmato 772  
tabelle di caratteri  
creazione 450  
impostazione 449  
valori di taglio 449  
tag ID3 656  
target  
contenuto caricato 84  
e area di validità 86  
Tasti di scelta rapida Esc 57  
tasti funzione, valori dei codici tasto ASCII 847  
tasti premuti, rilevamento 621  
tastiera  
tasti di scelta rapida per gli script bloccati 64  
valori ASCII 844  
tastierino numerico, valori dei codici tasto ASCII 846  
TCP/IP, connessione  
con l'oggetto XMLSocket 717  
invio di informazioni 690  
terminologia, ActionScript 869  
testo  
caricamento e visualizzazione 428, 430, 432  
definizione 876  
terminologia 415  
testo con antialiasing  
creazione di tabella 450  
informazioni 443  
limitazioni 444  
modifica di nitidezza e spessore 453  
proprietà antiAliasType, impostazioni 446  
sharpness, proprietà 453  
supporto 444  
supporto Flash Player 443  
thickness 453  
uso 447  
valore advanced 446  
valore normal 446

testo di input 417  
testo dinamico 417  
testo statico 417  
testo  
    Vedere anche campi di testo  
text  
    assegnazione di un testo a un campo di testo in fase di runtime 419  
    codifica 62  
    scorrevole 491  
    uso del tag <img> per applicare un flusso al testo intorno a immagini 479  
this, parola chiave 86, 642  
    area di validità 237  
    come prefisso 815  
    e area di validità 86  
    nelle classi 237  
    uso 811  
tipi di adattamento alla griglia, uso 456  
tipi di dati  
    annotazioni 350  
    assegnazione 346  
    assegnazione automatica 344  
    Boolean 337  
    complessi 335  
    conversione 334  
    definizione 334, 871  
    determinazione del tipo 349  
    di base 334, 335  
    e valori 203  
    MovieClip 339  
    null 340  
    Number 341  
    Object 341  
    Stringa 342  
    undefined 344  
    void 344  
tipizzazione forte 345, 350  
tipizzazione forte dei dati 350  
tipo di dati complesso (valore di dati) 335  
tipo di dati di base (valore di dati) 335  
Tipo di dati MovieClip, definizione 339  
tipo di dati non definito 344  
tipo di dati nullo 340  
tipo di dati Void 344  
Tipo MIME 679  
Transition, classe  
    animazione del livello di luminosità 574  
TransitionManager, classe  
    mediante API di disegno 604  
    mediante classe Tween 543  
transizioni  
    aggiunta mediante ActionScript 531  
    aggiunta mediante comportamenti 528  
    definizione 530  
trascinamento di clip filmato 390  
trasferimento di variabili tra filmato e server 697  
trasparenza e applicazione dell'effetto maschera 411  
tratti  
    impostazione dei parametri 599  
    Impostazione di stili 598  
Tween, classe  
    \_alpha, proprietà 543  
    animazione del livello di luminosità 574  
    animazione di filtri sfocatura 582  
    continueTo(), metodo 540, 543  
    dissolvenza oggetti mediante 538  
    e andamento 527  
    importazione 537  
    impostazione della durata dei fotogrammi 538  
    informazioni 527, 535  
    mediante API di disegno 604  
    mediante classe TransitionManager 543  
    onMotionFinished, gestore di eventi 541  
    per attivare animazione completata 539  
    uso 531, 537  
    yoyo(), metodo 541, 543

## U

UCS (Universal Character Set), definizione 493  
Unicode  
    codice di carattere 492  
    comando Prova filmato 772  
    definizione 493  
    supporto 61  
Universal Character Set, (UCS) 493  
URL, codifica per l'invio di informazioni 690  
UTF-8 (Unicode) 61, 493

## V

valore letterale di stringa 875  
valore letterale oggetto 142  
valori  
    e tipi di dati 203  
    gestione nelle espressioni 143  
valori letterali, definizione 873  
variabile della linea temporale, informazioni su 364

variabile locale, informazioni su 365  
variabili  
assegnazione di un nome 52  
assegnazione di valori 353  
caricamento 367, 372  
caricamento da file di testo esterno 431  
caricamento in campi di testo 427  
con codifica URL 367  
confronto definite e non definite 358  
conversione in XML 712  
definizione 350, 876  
dichiarazione 352  
e area di validità 362  
e l'elenco di controllo debug 777  
e operatori 355  
e scheda delle variabili di debug 776  
evitare conflitti tra nomi 422  
impostazione utilizzando un percorso 85  
instance 248  
invio agli URL 617  
linea temporale 364  
locale 365  
modifica di valore 354  
modifica nel Debugger 777  
passaggio da HTML 428  
passaggio di valori da stringa URL 367  
passaggio per riferimento 360  
regole per l'assegnazione dei nomi e linee guida 356  
trasferimento tra clip filmato e server 697  
uso 359  
uso di FlashVars per passare 370  
uso in un progetto 373  
uso in un'applicazione 357  
variabili predefinite 352  
variabili globali 363  
variabili URL, informazioni su 367  
variabili, globali 363  
verifica del tipo  
definizione 348  
dynamic 349  
esempio 348  
video  
aggiunta della funzionalità di ricerca 674  
configurazione del server per FLV 679  
creazione di file FLV 660  
creazione di un banner 663  
creazione di un oggetto video 661  
creazione di una barra di avanzamento per caricare  
file FLV 686  
cue point 667  
e Macintosh 680  
informazioni 659  
informazioni sui file FLV esterni 660  
lavoro con i cue point 670  
metadati 677  
navigazione di un file FLV 674  
precaricamento 665  
ricerca del cue point 675, 676  
ricerca di una durata specifica 674  
riproduzione di file FLV in fase di runtime 662  
tracciamento dei cue point 668  
utilizzo del gestore onMetaData 677  
Video Flash  
*Vedere* video  
Video FLV. *Vedere* video  
video, alternativa all'importazione 660  
virgolette, nelle stringhe 343  
volume, creazione del controllo scorrevole 626

## W

while 126  
with (istruzione) 820

## X

XML 709  
caricamento e visualizzazione di testo 432  
conversione variabile di esempio 711  
DOM 709  
esempio di utilizzo degli stili 473  
gerarchia 709  
in script sul lato server 711  
invio di informazioni con metodi XML 690  
invio di informazioni tramite socket TCP/IP 690  
XML Localization Interchange File Format 495  
XMLSocket, oggetto  
controllo dei dati 691  
loadPolicyFile 765  
metodi 717  
uso 717

