

ACCADEMIA DI BELLE ARTI DI URBINO

Anno accademico 2006-2007

Sistemi e linguaggi di programmazione avanzati

Guida di base alla programmazione

SISTEMI E LINGUAGGI DI PROGRAMMAZIONE AVANZATI

Guida alla programmazione

a cura di Bruno Migliaretti

Sommario

- 4 LA RAPPRESENTAZIONE DIGITALE DELLE INFORMAZIONI
Come le informazioni vengono codificate per poter essere elaborate da un calcolatore elettronico.
 - 15 LA COMPRESSIONE DEI DATI
Ridondanza delle informazioni e tecniche di compressione
 - 27 LA PROGRAMMAZIONE
Linguaggi e tipologie d'applicazione
 - 33 ALGORITMI
Definizione di un algoritmo, sua rappresentazione ed esecuzione
 - 40 IMPARIAMO AD INDENTARE
Come scrivere un codice sorgente chiaro e interpretabile da altri
 - 42 INTRODUZIONE ALLA LOGICA
L'algebra di Boole
 - 48 GLI ELEMENTI DEL LINGUAGGIO
Parole chiave, operatori, costanti
 - 55 VARIABILI E TIPI DI DATI
Cosa sono e come possiamo utilizzarle
 - 58 LA PROGRAMMAZIONE CONDIZIONALE
I costrutti per creare delle condizioni all'interno del programma: IF e ELSE IF
-

61	LA PROGRAMMAZIONE ITERATIVA	La programmazione iterativa: WHILE, DO e FOR
67	FUNZIONI E METODI	Semplifichiamo il nostro lavoro di progettazione con le funzioni
80	LE CLASSI	I vantaggi della programmazione orientata agli oggetti.
87	OBJECT, ARRAY E DATE	Esempi di come funzionano alcune classi predefinite.
102	LA CREAZIONE DI CLASSI	Come creare classi personalizzate.
131	DISEGNAMO UN RETTANGOLO	Creiamo insieme una classe e utilizziamola in un'applicazione.
144	L'EREDITARIETA	Creazione di sottoclassi e riutilizzo del codice.
153	APPENDICI	

Rappresentazione digitale delle informazioni

Definizione di informazione

In termini di economia del linguaggio possiamo affermare che informazione è tutto ciò che riduce incertezza. Esempi: una foto riduce l'incertezza di chi la osserva riguardo alla scena raffigurata, l'espressione del volto di un interlocutore riduce l'incertezza sul suo stato d'animo, un fruscio riduce l'incertezza sul luogo in cui cercare un fagiano, ogni parola riduce l'incertezza sul significato di una frase.

È una definizione limitata (in termini di antropologia culturale sappiamo che non sempre ciò che aggiunge sapere sottrae incertezza, anzi...) ma che ci aiuta ad introdurre la definizione successiva: la definizione di dato.

Per dato intendiamo un'informazione codificata. Quando l'informazione viene codificata in un linguaggio convenzionale per poter essere rappresentata, scambiata, memorizzata ed elaborata otteniamo un dato.

La codifica delle informazioni può rendere omogenea la rappresentazione e l'elaborazione di informazioni di natura diversa (suoni, immagini, testi, numeri) e renderli disponibili per l'elaborazione automatica.

Informatica significa informazione automatica. Più in particolare diciamo che l'informatica è quella disciplina che studia l'elaborazione automatica delle informazioni. Condizione prima perché le informazioni possano essere elaborate e che vengano codificate in un linguaggio convenzionale che l'informatica possa elaborare.

Ora vediamo di comprendere meglio il concetto di automazione. Per automatico si intende tutto ciò che compie un compito prestabilito senza l'intervento umano. Quindi quando si parla di *automazione* si fa riferimento alla realizzazione di strumenti per la soluzione di problemi o l'esecuzione di processi in maniera indipendente dall'attività umana.

In termini di economia dei processi un qualche processo viene automatizzato quando il numero di volte che esso deve essere eseguito è

sufficientemente grande da rendere conveniente la progettazione e la costruzione di un sistema automatico che lo risolva. Gli stessi problemi che vengono trattati in maniera automatica sono stati originariamente trattati dall'uomo.

Alcuni esempi

Problema originale (non ripetitivo)	Problema sistematico e ripetitivo
<i>Soluzione manuale</i>	<i>Soluzione automatica</i>
Determinare il diametro di una pallina	Classificare un insieme di palline in base al diametro
<i>Calibro</i>	<i>Griglie forate</i>
Scrivere una lettera	Riprodurre un testo in migliaia di copie
<i>Carta e penna</i>	<i>Stampa tipografica</i>
Regolare il traffico in caso di emergenza	Regolare il traffico ad ogni incrocio
<i>Vigile urbano</i>	<i>Semaforo</i>
Riempire un contenitore d'acqua	Riempire ripetutamente un contenitore con la stessa quantità d'acqua
<i>Rubinetto manuale</i>	<i>Rubinetto a galleggiante</i>

Che cosa è un computer

Un elaboratore elettronico è una macchina (elettronica) programmabile per l'elaborazione automatica di dati. Un elaboratore è costituito da una parte così detta Hardware (componenti tangibili e permanenti di un computer). E da una parte software (insieme dei programmi, componenti intangibili e temporanei, che controllano il funzionamento dell'hardware).

L'hardware è in grado di eseguire un insieme finito di operazioni elementari e di interpretare istruzioni in un linguaggio predefinito. Il software specifica sequenze di istruzioni da eseguire per arrivare alla soluzione di un problema.

Un computer è programmabile. È in grado, cioè, di svolgere compiti diversi in base ad un programma. Un elaboratore può essere utilizzato ripetutamente per risolvere problemi diversi o istanze diverse dello stesso problema. In molti casi, l'automazione di un processo è conveniente solo se si può realizzare con elaboratori programmabili già progettati e sviluppati per altre applicazioni. In tal caso solo il programma deve essere

riprogettato. Il processo di codifica consente di utilizzare lo stesso elaboratore per manipolare informazioni diverse. Ogni informazione può (a meno di un'approssimazione) essere rappresentata come una sequenza di due soli simboli (di solito 0 e 1) che nel campo informatico vengono chiamati BIT.

La parola BIT nasce dalla contrazione delle parole inglesi Binary Digit e letteralmente BIT significa "cifra binaria". Cioè una cifra che può assumere solo due valori (di solito si parla di 0 e 1). La rappresentazione fisica di un bit si può ottenere da un qualunque sistema in grado di trovarsi in due stati diversi mutuamente esclusivi. Ad esempio un bit può essere rappresentato da un interruttore che può trovarsi nei due stati acceso/spento. L'interruttore infatti si trova alla base del funzionamento dei circuiti logici che costituiscono le unità di elaborazione dei microprocessori.

Allo stesso modo un bit può essere rappresentato da un condensatore che si trova nei due stati carico/scarico. Tutte le memorie principali così dette dinamiche (come la RAM) si basano su questo principio, memorizzando i singoli bit in piccolissimi condensatori in modo tale da rappresentare un 1 se il condensatore è carico e uno 0 se questo è scarico.

D'altra parte, nelle memorie di massa non volatili come i dischi fissi, i bit vengono memorizzati attraverso particelle magnetiche orientabili. In questo modo per rappresentare un bit di valore 1 si orienta una particella in una direzione e per rappresentare uno 0 si orienta nel verso opposto.

La codifica delle informazioni ci consente, cioè, di utilizzare determinati supporti fisici (hardware) per rappresentarle.

Codifica delle informazioni

La codifica delle informazioni in termini di bit è l'operazione che consente di utilizzare i calcolatori elettronici per elaborare dati di qualsiasi natura. Abbiamo già detto che un bit può rappresentare solamente due diverse informazioni (1 o 0, acceso o spento, ecc.). Per codificare insieme con un numero maggiore di informazioni/stati usiamo stringhe di bit di lunghezza variabile. Una stringa è una sequenza di n bit e viene anche detta parola dall'inglese word. Con un numero n bit si possono formare 2^n configurazioni diverse e quindi si possono rappresentare 2^n informazioni diverse. La codifica è una convenzione che associa un significato ad ogni configurazione di bit.

I calcolatori sono oggetti finiti che elaborano e memorizzano un numero finito di bit. Un numero finito di bit permette di codificare un numero finito di informazioni diverse. Se l'insieme delle informazioni da rappresentare è superiore alle combinazioni di BIT che ho a disposizione non avrò una rappresentazione esatta delle informazioni ma una rappresentazione approssimata.

Un insieme di informazione può essere limitato o illimitato (infinito) e può essere discreto a continuo. Facciamo alcuni esempi.

L'insieme dei caratteri di una lingua è limitato e discreto in quanto composto da un numero definito di segni.

L'insieme dei numeri interi è infinito e discreto: la sequenza dei numeri è infinita, ma, dati due valori qualsiasi, il numero degli elementi compresi tra i due valori è sempre finito.

L'insieme dei numeri reali è invece infinito e continuo: la sequenza dei numeri reali è infinita e il numero di elementi compresi tra due elementi qualsiasi della sequenza anche molto vicini tra loro è sempre infinito.

L'insieme dei colori è limitato e continuo: la sensibilità del mio occhi limita la sequenza dei colori dal rosso al violetto, ma tra due elementi della sequenza potrò sempre immaginare un numero infinito di sfumature.

La strategia adottata nella codifica cambia a seconda delle caratteristiche dell'insieme di informazioni da rappresentare.

- Normalmente gli insiemi finiti e discreti (come i caratteri di una lingua) vengono rappresentati esattamente.
- Gli insiemi illimitati vengono limitati. Nella convenzione che definisce la codifica si stabilisce che venga rappresentato un sottoinsieme di informazioni la cui dimensione dipende dal numero di bit che si dedicano alla codifica (gli elementi del sottoinsieme sono rappresentati esattamente, gli altri non sono rappresentati).
- Gli insiemi continui vengono partizionati. Normalmente l'insieme viene suddiviso in intervalli di uguale ampiezza e viene codificato come se fosse un insieme discreto.

Gli insiemi illimitati e continui saranno sia limitati che partizionati.

Codifica del Testo

Un testo è una sequenza di caratteri alfabetici, separatori e caratteri speciali come punteggiatura, accentuazioni, ecc. La codifica ASCII (American Standard Code for Information Interchange) prevede l'uso di 128 caratteri diversi. Ogni carattere è associato ad una diversa configurazione di 7 bit.

La codifica ASCII si è piuttosto limitata. Modellata sulle lingua inglese non comprende le lettere con modificatori quali accenti, dieresi, tilde ecc. Per queste limitazioni sono nate estensioni non standard del codice ASCII che utilizzano 8 bit (1 byte) per carattere, portando a 256 il numero di caratteri disponibili. Si sono diffuse quindi diverse tabelle di caratteri (specializzate

per gruppi di lingue) che hanno in comune i primi 128 caratteri (ASCII standard) e usano i rimanenti per offrire alle lingue di riferimento tutti i caratteri necessari. Questo processo ha reso la codifica dei caratteri relativa. Il codice di un carattere non è univoco ma dipende dalla tabella che viene impiegata. Ad esempio in un computer che esegue Microsoft Windows utilizzando la tabella codici predefinita 1252 e in un computer Apple Macintosh che utilizza la tabella codici Macintosh Roman, la posizione 232 di queste tabelle dei caratteri corrisponde rispettivamente alla lettera **è** nel sistema Windows ed alla lettera **Ê** in quello Macintosh.

Con l'entrata delle lingue asiatiche nel mondo dei personal computer è nata l'esigenza di aumentare in maniera sensibile il numero dei segni da codificare. Nasce quindi due progetti col fine di costruire un'unica tabella universale dei caratteri, il progetto ISO 10646 e il progetto Unicode.

Lo standard ISO10646/Unicode si basa su una codifica a 32 bit che consente oltre due miliardi di possibili caratteri, numero che dovrebbe essere decisamente sufficiente per realizzare una tavola di codifica dei caratteri veramente universale.

L'implementazione informatica di Unicode tuttavia comporta dei problemi, in particolar modo per quanto riguarda la crescita della dimensione dei files, per tal motivo sono stati creati dei charset detti UTF, *Universal Character Set Transformation Format*.

UTF usa una tecnica di bit-shifting (spostamento dei bit) per codificare i caratteri Unicode. In sostanza l'UTF usa 7 bit per carattere per codificare i primi 127 caratteri corrispondenti all'ASCII standard, e attiva l'ottavo bit solo quando serve la codifica Unicode. Ogni carattere Unicode viene quindi codificato con un numero variabile di byte che va da 1 a 3. Il vantaggio è che il testo è visualizzabile con un normale editor di testi, anche se naturalmente i caratteri diversi dall'ASCII standard potrebbero apparire non correttamente. Per i testi comuni utilizzati in occidente consente di mantenere più compatte le dimensioni dei files rispetto ad UCS in quanto la maggior parte dei caratteri sono ASCII. Inoltre, l'UTF-8 attraversa indenne, come se fosse un normale testo, anche i sistemi e i programmi che non supportano Unicode. Per questi motivi l'UTF è molto usato nella posta Internet ed in alcuni sistemi operativi.

Tra le codifiche UTF la più comune è la UTF-8, che tra l'altro è la codifica di default di XML.

Indipendentemente dalla codifica usata un testo è rappresentato dalla sequenza di byte associati ai caratteri che lo compongono, nell'ordine in cui essi compaiono. Un testo di 1000 caratteri richiede 1000 byte (1 Kb) se viene utilizzata una tabella di testo ANSI, 4000 byte (4 Kb) se viene utilizzata la codifica Unicode e un numero variabile di byte, che dipende dai caratteri codificati se si usa una codifica UTF.

Codifica dei numeri

Per la rappresentazione dei numeri è particolarmente diffusa la codifica binaria, che fornisce una diretta trasposizione in un alfabeto a due valori della notazione decimale da noi comunemente utilizzata. Il numero di simboli dell'alfabeto è detto base di numerazione. La codifica binaria e quella decimale utilizzano basi diverse (2 e 10, rispettivamente) ma sono entrambe notazioni posizionali, in quanto permettono di rappresentare ogni numero come sequenza di cifre del tipo:

$$c_n \ c_{n-1} \ \dots \ c_1 \ c_0 . c_{-1} \ c_{-2} \ \dots \ c_{-m}$$

Il valore (v) del numero rappresentato è la somma pesata delle cifre, dove il peso di ogni cifra dipende dalla base di numerazione (B) e dalla posizione della cifra:

$$v = c_n B^n + c_{n-1} B^{n-1} + \dots + c_1 B + c_0 + c_{-1} B^{-1} + c_{-2} B^{-2} + \dots + c_{-m} B^{-m}$$

Es: $(101.1)(\text{base } 2) = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} = (5.5)(\text{base } 10)$.

Numeri interi.

I numeri interi sono un insieme discreto illimitato. Per poter essere codificati devono essere limitati. In genere ci si restringe ad un sottoinsieme simmetrico rispetto allo 0. Avendo a disposizione n bit, se ne usa uno per rappresentare il segno e i restanti $n-1$ per rappresentare il modulo. Se $a = n-1$ (il numero dei bit a disposizione meno uno) il massimo numero rappresentabile è (in modulo) 2^{a-1} . Se il risultato di un'operazione superiore a 2^{a-1} o inferiore a -2^{a-1} non può essere codificato e il calcolatore restituisce un messaggio di overflow.

Numeri reali.

I numeri reali sono un insieme continuo e illimitato. Per poterli rappresentare occorre limitarli (in modo simmetrico rispetto allo 0) e partizionarli. Esistono due tipi di rappresentazione dei numeri reali: rappresentazione in virgola fissa e rappresentazione in virgola mobile o *floating-point*.

Rappresentazione in virgola fissa.

Nella rappresentazione a virgola fissa un bit viene serve a indicare il segno (+ o -) un certo numero di bit (diciamo a) vengono dedicati a rappresentare le cifra prima della virgola e i rimanenti (diciamo b) le cifre dopo la virgola. Più a sarà grande più potremo rappresentare numeri positivi grandi e numeri negativi piccoli, più b sarà grande più aumenterà la precisione. Il difetto della rappresentazione dei numeri reali in virgola fissa è la rigidità.

Rappresentazione in virgola mobile (floating-point).

Il numero da rappresentare viene espresso nella forma

$$S \ 0.M \ B^{seE}$$

dove s rappresenta il segno, M la mantissa, E l'esponente, se il segno dell'esponente e B la base di numerazione. Non tutti i numeri hanno rappresentazione univoca. Questo comporta perdita di efficienza e difficoltà di confronto. Quando esistono rappresentazioni equivalenti dello stesso numero, tra queste si definisce normale quella con l'esponente più piccolo. Da una rappresentazione non normale a quella normale equivalente si passa traslando verso sinistra le cifre della mantissa e sottraendo 1 all'esponente.

Codifica delle immagini

Le immagini sono informazioni continue in tre dimensioni: due spaziali ed una colorimetrica. Per codificarle occorre operare tre discretizzazioni. Le due discretizzazioni spaziali riducono l'immagine ad una matrice di punti colorati, detti pixel. La terza discretizzazione limita l'insieme di colori che ogni pixel può assumere.

Immagini bitmap

Il modo più elementare per rappresentare un'immagine consiste nel rappresentare il colore di ogni pixel. Il numero di bit necessari a rappresentare ogni pixel dipende dal numero di diversi colori disponibili. Un byte consente di rappresentare 256 colori. Un'immagine a 256 colori è una sequenza di byte.

Un'immagine di 100X100 pixel a 256 colori richiede 10000 byte (10 Kb) per essere rappresentata. Un'immagine a due soli colori (ad esempio bianco e nero) richiede un solo bit per pixel. Le immagini codificate secondo questa tecnica vengono dette anche immagini raster o immagini bitmap. Questo tipo di immagini vengono solitamente usate per fotografie, scansioni e in genere per tutte le immagini che prevedono sottili gradazioni nei colori e nelle forme.

Le immagini raster sono caratterizzate da tre informazioni principali che ne consentono la corretta interpretazione: dimensioni, numero dei colori, risoluzione.

Le dimensioni consentono di ricostruire correttamente l'immagine collocando i pixel al loro posto.

Il numero dei colori è definito dal numero di bit che viene dedicato alla descrizione del colore di ogni pixel:

- immagini monocromatiche: un bit per ogni pixel (se il bit è 0 è bianco, se no nero)

- immagini a 256 colori: 8 bit (un byte) per ogni pixel. A ogni immagine è associata una tabella di 256 colori (detta tavolozza o palette) in cui sono descritti i contenuti RGB (Red, Green, Blue) di ognuno dei 256 colori usati. In altre parole i 256 sono numerati da 0 a 255 e l'elaboratore, quando decodifica l'immagine controlla a che colore corrisponde il numero che descrive ogni pixel. In questa maniera si possono avere in ogni immagine al massimo solo 256 colori diversi ma scelti
- immagini a 16 bit: vengono riservati 16 bit (2 byte) per descrivere il colore di ogni pixel. Ogni colore è descritto dalle sue componenti RGB (Red, Green, Blue). Vengono messi a disposizione 5 bit per ogni componente (l'ultimo bit non è usato). Avremo cioè colori descritti con 2^5 (cioè 32) livelli di Rosso, Verde e blue per un totale di 32.768 colori.
- immagini a 24 bit: vengono riservati 24 bit (3 byte) per descrivere il colore di ogni pixel. Ogni colore è descritto dalle sue componenti RGB (Red, Green, Blue). Vengono messi a disposizione 8 bit (un byte) per ogni componente. Avremo cioè colori descritti con 2^8 (cioè 256) livelli di Rosso, Verde e blue per un totale di 16.777.216 colori.
- immagini a 32 bit: vengono riservati 32 bit (4 byte) per descrivere il colore di ogni pixel. Come nelle immagini a 24 bit vengono messi a disposizione 8 bit (un byte) per ogni componente. Il quarto byte descrive 256 livelli di trasparenza (il cosiddetto canale alpha) che può avere il pixel.

Infine la risoluzione definisce il rapporto che c'è tra la dimensione dell'immagine in pixel e la dimensione reale che l'immagine ha quando viene resa da un dispositivo (stampante, fax, schermo, ecc). La risoluzione di misura normalmente in punti (pixel) per pollice o in punti per centimetro.

Immagini vettoriali

La grafica vettoriale scompone l'immagine in gruppi logici di componenti (linee, cerchi, rettangoli, ecc.) e memorizza le forme in termini di coordinate e colori dei vari elementi geometrici che le compongono. In definitiva un file che memorizza un'immagine di tipo vettoriale è costituito da un'insieme di coordinate e dalle informazioni sul colore di ciascuna forma. Nel momento della visualizzazione i punti descritti dalle coordinate vengono disegnati e colorati a dovere. La grafica vettoriale viene comunemente usata nel disegno animato, nella grafica lineare e negli strumenti CAD.

Un grande vantaggio delle immagini vettoriali rispetto a quelle raster/bitmap sta nella minor dimensione del file che le memorizza e nella capacità di essere variate di dimensioni senza subire alcuna distorsione. Le

immagini bitmap, invece, se ridimensionate rispetto alle dimensioni originali di acquisizione, hanno la tendenza a perdere di risoluzione risultando distorte o sfocate.

Filmati video (immagini in movimento). I filmati video possono essere considerati sequenze di fotogrammi. Abbiamo già visto come codificare immagini, e quindi sappiamo come codificare ogni fotogramma. Il numero di fotogrammi nell'unità di tempo determina la qualità del movimento. E' stato sperimentato che l'occhio umano non percepisce discontinuità nel movimento al di sotto del venticinquesimo di secondo. Pertanto, una frequenza di riproduzione dei fotogrammi superiore a 25Hz sarebbe superflua se lo scopo fosse la riproduzione di un video. D'altro canto potrebbe non esserlo se servisse a documentare un esperimento di brevissima durata o se il video dovesse essere rivisto al rallentatore.

Es: Filmato di 10 minuti, con risoluzione di 100x100 pixel a 256 colori: dimensione complessiva di $600 \times 25 \times 100 \times 100 \times 8 = 1.2\text{Gbit}$.

Codifica di segnali analogici

Un segnale analogico è una grandezza fisica che varia nel tempo e che può assumere un insieme continuo (e quindi infinito) di valori. Un segnale digitale invece può assumere solo un insieme discreto (e generalmente finito) di valori. Un segnale analogico ha anche la proprietà di essere tempo-continuo: cioè il suo valore è significativo (e può variare) in qualsiasi istante di tempo.

Diversamente, un segnale digitale viene anche detto segnale tempo-discreto: poiché il suo valore ha interesse solo in istanti di tempo prestabiliti, generalmente equidistanziati, mentre non ha alcun valore tra un istante ed un altro.

La rappresentazione più naturale di un segnale s è la sequenza dei suoi valori istantanei $s(t)$. Ci poniamo prima il problema di rappresentare un valore istantaneo, e poi quello di rappresentarne la sequenza.

Il valore istantaneo di un segnale può essere rappresentato da un numero, e come tale può essere codificato utilizzando notazione in virgola fissa o in virgola mobile. Tali codifiche inducono limitazioni e partizionamenti sullo spazio dei possibili valori che il segnale può assumere (quantizzazione del segnale) .

Una volta codificato il valore istantaneo del segnale si ottiene la sequenza temporale ripetendo la codifica ad intervalli prestabiliti. In generale, se il segnale da codificare era tempodiscreto, la scelta naturale è quella di rappresentare il valore del segnale in tutti e soli gli istanti in cui esso è significativo. Per la codifica dei segnali analogici, che come abbiamo detto sono tempocontinui occorre operare una discretizzazione dell' intervallo

temporale per ottenere una rappresentazione finita. L'operazione di discretizzazione più comune è il campionamento, che consiste nell'osservare il segnale tempo-continuo ad istanti di tempo prefissati (generalmente equidistanziati) in modo da ottenere un segnale tempodiscreto che assume lo stesso valore di quello originale nei punti di campionamento.

In definitiva, le operazioni di campionamento e quantizzazione trasformano un segnale analogico tempocontinuo in un segnale digitale tempodiscreto.

Codifica del suono

Il suono è un segnale analogico tempo-continuo. Ci sono diversi modi per descrivere la natura di tale segnale. Per fissare le idee pensiamo che il suono prodotto da un altoparlante è prodotto dalla vibrazione di una membrana. Descrivendo la posizione della membrana nel tempo (e quindi il suo spostamento) a tutti gli effetti descriviamo il suono. Assumiamo allora che il segnale analogico $s(t)$ esprima la posizione della membrana (rispetto ad un riferimento fissato) all'istante di tempo t . Tale segnale deve essere campionato e discretizzato per poter essere codificato in termini di bit. Per valutare la qualità della codifica possiamo pensare di usare il segnale quantizzato e tempo-discreto (quello cioè effettivamente rappresentato dalla nostra codifica) per muovere la membrana di un altoparlante. Se il campionamento è troppo rado e vengono usati pochi bit per codificare ogni valore istantaneo, la perdita di informazione degrada la qualità del suono (il suono riprodotto è sensibilmente diverso da quello originale).

Per quanto riguarda il campionamento, è dimostrato che il suono percepibile dall'orecchio umano viene riprodotto fedelmente se la frequenza di campionamento (il numero di campioni in un secondo) è non inferiore a 30KHz. Per rendere intelligibile il parlato è sufficiente una frequenza di campionamento di 8KHz. La quantizzazione introduce comunque una distorsione. Questa tecnica di codifica del suono è quella comunemente usata nella codifica di tipo wave che troviamo nei file .wav. Questa codifica tenta di ricostruire fedelmente la forma d'onda del suono campionandola adeguatamente. Il file che si ottiene è normalmente di grandi dimensioni.

Es: Telefono. La comunicazione telefonica deve riprodurre la voce umana ai due estremi del collegamento garantendo la comprensione del parlato. A tal fine lo standard prevede un campionamento a 8KHz ed una quantizzazione a 256 livelli (codificati con 8 bit). Il segnale digitale che ne risulta usa 64Kbit per codificare ogni secondo di conversazione. 10 minuti di conversazione richiedono 38.4Mbit.

La codifica di tipo MIDI, invece ha tra i formati audio lo stesso ruolo che il formato vettoriale ha tra i formati grafici. Infatti, il formato MIDI acronimo di Musical Instrument Digital Interface, non è una registrazione sonora ma

contiene dei comandi da impartire ad un sintetizzatore sonoro (software o hardware) che riproduce il suono di uno strumento musicale. I file MIDI per questo motivo sono molto piccoli ed hanno il formato del tipo:

suona la nota x per un tempo y con lo strumento z.

La codifica MIDI, quindi riproduce un suono in maniera sintetica. Le nuove suonerie polifoniche dei telefoni cellulari di ultima generazione si basano su una codifica di tipo MIDI.

La compressione dei dati

Spesso risulta utile ridurre le dimensioni dei dati su si sta lavorando in modo da renderne più agevole l'archiviazione e la trasmissione. Una condizione tipica in cui viene applicata la compressione sui dati è la ridondanza. In poche parole, dei dati che contengono le stesse informazioni in qualche modo ripetute (ridondanza) sono facilmente comprimibili. Ad esempio, quando ci si trova di fronte a un testo scritto si incorre in esempi di ridondanza lampanti. Si consideri, per esempio, l'uso che la lingua italiana fa della lettera "q" che è sempre seguita da una "u": nessuno avrebbe difficoltà nel capire il significato della parola "q*adro'", nonostante essa non sia scritta correttamente. Quindi, una prima tecnica di compressione della lingua italiana potrebbe essere basata sull'omissione del carattere "u" tutte le volte che questo è preceduto da un carattere "q".

La presenza di ridondanza in una lingua, in un testo scritto, o in una qualunque stringa di simboli non è però nefasta, anzi, essa svolge il compito fondamentale di rendere robusto il messaggio contenuto in quella stringa. Per robusto si intende facilmente comprensibile e non incline ad essere male interpretato anche nel caso in cui il messaggio venga trasmesso in modo solo parziale. Errori di trasmissione su codici ridondanti sono molto più facilmente correggibili.

D'altra parte, un testo che sia stato compresso al massimo è molto fragile, in quanto, per definizione di massima compressione, ciascun simbolo sarà portatore di informazione, e quindi una sua alterazione porterà ad una perdita irrimediabile di quella stessa informazione.

In definitiva, l'uso di codifiche ridondanti può avere due motivazioni: la flessibilità e l'affidabilità.

- La flessibilità indica la possibilità di utilizzare la stessa codifica in situazioni diverse. Es: il cosiddetto "baco del millennio" (millennium bug) che costa al mondo migliaia di miliardi è dovuto alla scarsa lungimiranza con cui i produttori di software hanno deciso di utilizzare due sole cifre decimali per rappresentare l'anno nelle date, dando per scontate (e quindi prive di contenuto informativo) le prime 2, fino ad ora sempre uguali a 19.

- L'affidabilità deriva dalla capacità di alcune codifiche ridondanti di rivelare o correggere errori. Un codice non ridondante associa univocamente una configurazione ad un significato. Un errore nella configurazione comporta un errore di interpretazione del significato. Una codifica ridondante a rivelazione d'errore associa significati utili solo ad un sottoinsieme delle configurazioni possibili. Se un errore trasforma una configurazione significativa in una non significativa, chi la interpreta riconosce la presenza dell'errore. Una codifica ridondante a correzione d'errore associa molte configurazioni allo stesso significato. Se un errore trasforma una configurazione in una a cui è associato lo stesso significato, l'interpretazione resta corretta.

Codifiche ridondanti si prestano a compressione, come dimostrano gli esempi precedenti, ma esistono anche tecniche di compressione che agiscono su dati irridondanti sfruttando proprietà tipiche dell'informazione da rappresentare.

Tecniche generiche di compressione

Per quanto riguarda le tecniche di compressione una prima distinzione che bisogna fare è tra compressione lossy e lossless. La prima tecnica effettua una compressione dei dati definita " con perdita d'informazione" o anche distruttiva in quanto, una volta applicata questa tecnica, non è più possibile ricostruire in maniera esatta i dati di partenza attraverso il processo di decompressione.

In definitiva, c'è stata, una perdita irrimediabile di informazione.

Le tecniche di tipo lossy sono legate a specifiche codifiche delle informazioni (immagini, suoni, filmati, ecc.) e sfruttano le loro specifiche caratteristiche. In questi casi perdita di informazione significa perdita di qualità.

Questa tecnica è molto utilizzata nel campo della compressione dei formati multimediali soprattutto laddove è spesso inutile ricostruire, come nel caso delle fotografie, l'informazione iniziale con una risoluzione maggiore di quella che l'occhio umano può apprezzare o di quella che uno specifico mezzo (ad esempio il video) può restituire.

Le tecniche di compressione di tipo lossless permettono invece di recuperare interamente l'informazione contenuta nel documento codificato prima della sua compressione, ma la loro efficienza di compressione è minore. Queste tecniche sono comunque le uniche utilizzabili in quei casi in cui non è possibile accettare neppure la minima perdita delle informazioni. Ad esempio la tecnica di compressione che sta alla base dei compressori tipo Winzip o Winrar è una tecnica di tipo lossless in quanto non sarebbe accettabile perdere dell'informazione comprimendo ad esempio un file di installazione di un programma o un file di testo.

Tecniche di tipo lossless

Compressione run-length

La compressione di tipo run-length fa parte delle tecniche di compressione di tipo lossless cioè senza perdita di dati. Essa si basa su un algoritmo denominato di *Run Length Encoding* (RLE) che è uno dei più semplici algoritmi di compressione mai progettati. Si basa sul fatto che nei dati da comprimere esistono sequenze, definite run, che si ripetono costantemente. Una volta individuate le sequenze ripetute, vengono sostituite da un unico simbolo e dal numero delle ripetizioni presenti.

Ad esempio data la stringa di bit 01110000 l'algoritmo RLE la comprimerebbe codificandola in 03*14*0 che sta ad indicare "0 tre volte 1 quattro volte 0" .

Esistono numerose varianti di RLE le cui principali differenze consistono nella lunghezza minima da attribuire ad un run. Nell' esempio appena descritto abbiamo usato una lunghezza del run pari a uno cioè si prendeva in considerazione un carattere alla volta e si cercava se questo veniva

ripetuto consecutivamente. Allo stesso modo si possono prendere in considerazione gruppi di simboli e cercare se l' intero gruppo viene ripetuto all' interno della stringa.

Compressione a codifica differenziale

In alcuni casi, le informazioni sono costituite da blocchi di dati, ognuno dei quali differisce leggermente dal precedente come, ad esempio, i fotogrammi successivi di un filmato. In questo caso sono utili le tecniche di compressione che utilizzano la codifica differenziale. L'approccio di queste tecniche è quello di memorizzare non il blocco stesso ma le sue differenze rispetto al precedente. È evidente che in questo caso se si dovesse corrompere un blocco compresso durante la trasmissione, risulterebbe compromessa la ricostruzione/decompressione di tutti i blocchi successivi.

Codifiche basate su dizionari

Il termine dizionario si riferisce all' insieme di elementi di base sui quali viene ricostruito il messaggio compresso. In pratica viene utilizzato un insieme di simboli (dizionario) per codificare un messaggio. I simboli del dizionario rappresentano particolari sequenze di bit e durante la compressione, ad ogni sequenza riconosciuta viene sostituito il simbolo corrispondente. La particolarità di queste tecniche sta nel fatto che il dizionario viene creato dinamicamente durante il processo di compressione. Tale tecnica sta alla base dell' algoritmo di compressione Lempel-Ziv che troviamo nella maggior parte dei tools di compressione quali WinZip.

Compressione delle immagini

Formato GIF

La sigla GIF è acronimo di Graphic Interchange Format. Questo tipo di compressione rientra nelle tecniche di tipo lossless, cioè senza perdita di dati. La caratteristica del formato GIF è che esso può esportare solo immagini che contengono al massimo 256 colori. Se l'originale contiene un numero più elevato di colori quindi, è necessario effettuare una riduzione e la perdita di qualità sarà significativa. Il formato GIF usa colori a 8 bit ed è efficace per comprimere immagini vettoriali, geometriche o testo. Questo formato fu diffuso negli anni Ottanta come metodo efficiente di trasmissione delle immagini su reti di dati. All'inizio degli anni Novanta i progettisti originali del web lo adottarono per l'efficienza che offriva. Oggi la stragrande maggioranza delle immagini sul web è in questo formato ed è supportato da tutti i browser web.

Il formato GIF usa una forma di compressione LZW che mantiene inalterata la qualità dell'immagine, ovvero riduce le dimensioni del file senza pregiudicare la qualità grafica dell'immagine. Questo formato è basato sull'uso di una tavolozza di colori: anche se il singolo colore può essere uno fra milioni di sfumature, solo un certo numero di essi è disponibile (al massimo $256=8\text{bit}$). I colori sono memorizzati in una 'tavolozza', una tabella che associa un numero ad un certo valore di colore.

La limitazione a 256 colori appariva ragionevole all'epoca della creazione del formato GIF perché non erano ancora diffusi dispositivi in grado di visualizzarne un numero superiore. Per disegni al tratto, fumetti, fotografie in bianco e nero sono di regola sufficienti 256 colori. Minore sarà il numero di colori presenti nell'immagine e maggiori saranno le possibilità di compressione, ovvero minori saranno le dimensioni del file in quanto sarà ridotta la dimensione della tavolozza e quindi il numero di codici per descrivere i colori.

Il formato GIF consente anche di salvare le immagini in un formato interlacciato. Il formato a interlacciamento produce una visualizzazione graduale di un'immagine in una serie di passate sempre più definite a mano a mano che i dati arrivano al browser. Ogni nuovo passo crea un'immagine più nitida fino al completamento dell'intera immagine.

Il formato GIF consente anche di definire un colore come trasparente. Nelle aree di colore contrassegnato come trasparente, verrà visualizzato il colore di sfondo. Questa proprietà viene utilizzata per le animazioni e per la sovrapposizione di più immagini.

Il formato GIF, infine prevede una serie di semplici comandi con cui si può stabilire la frequenza con cui il fotogrammi di una animazione sono visualizzati, se l'animazione è continua o viene ripetuta un numero finito di

volte, se le immagini visualizzate vengono sovrapposte a quelle precedenti o lo sfondo viene cancellato. In altre parole GIF è anche un formato programmabile.

Con l'evoluzione dei personal computer (maggiore risoluzione dei video, maggior numero di colori riproducibili), oggi si usano principalmente immagini a 16, 24 o 32 bit. Ciò nonostante, anche se in misura minore, il formato GIF è ancora molto utilizzato sia in quanto formato che consente animazioni programmate, sia perché, con determinate immagini risulta più efficiente del formato JPEG (oggi sicuramente il formato di compressione più utilizzato). In realtà oggi utilizzare il formato GIF significa perdita di informazione. Il processo di conversione avviene di fatto in due fasi: una prima fase (lossy) in cui l'immagine originale (normalmente a 24 o 32 bit) viene ridotta a 256 colori creando una palette ottimizzata per rappresentarla al meglio e una seconda fase (lossless) in cui all'immagine viene applicata la compressione GIF.

Molto spesso il formato GIF risulta particolarmente efficiente quando si convertono in bitmap disegni realizzati con programmi per disegno vettoriale con limitato numero di colori e uso di colori pieni.

Formato JPEG

Un formato grafico utilizzato più frequentemente sul Web per ridurre le dimensioni dei file grafici è lo schema di compressione JPEG (*Join Photographic Expert Group*). A differenza delle immagini GIF, le immagini JPEG sono policrome (24 bit, o 16,8 milioni di colori). Questo tipo di immagini ha generato un altissimo interesse tra fotografi, artisti, progettisti grafici, specialisti della composizione di immagini mediche, storici dell'arte e altri gruppi per i quali la qualità dell'immagine è d'importanza fondamentale e per i quali non è possibile accettare compromessi sulla fedeltà dei colori tramite retinatura di un'immagine a colori a 8 bit.

Una forma più recente di JPEG, chiamata JPEG progressivo, conferisce alle immagini JPEG la stessa gradualità di visualizzazione delle immagini GIF interlacciate; al pari di queste ultime, le immagini JPEG progressive impiegano spesso un tempo maggiore per lo scaricamento sulla pagina rispetto ai JPEG standard, ma offrono al lettore un'anteprima più rapida.

La compressione JPEG utilizza una sofisticata tecnica matematica, chiamata trasformazione discreta del coseno, per produrre una scala scorrevole di compressione delle immagini. Tale tecnica si basa su di una codifica dell'immagine percettiva in cui viene distinta la luminosità dei pixel dal loro colore. Il motivo di tale distinzione è che l'occhio umano è più sensibile alle variazioni di luminosità che non a quelle di colore. Lo standard base di JPEG trae vantaggio da questo fenomeno codificando ogni componente della luminosità, ma dividendo l'immagine in blocchi di quattro pixel e registrando solo il colore medio di ogni blocco. La rappresentazione finale

preserva dunque i cambiamenti di luminosità, attenuando i repentini mutamenti cromatici dell'immagine originale. Il vantaggio è che ogni blocco di quattro pixel è rappresentato soltanto da 6 valori (4 di luminosità e 2 di colore) anziché 12 valori che sarebbero necessari in un sistema a 3 valori per pixel (immagini a 24 bit RGB).

È possibile scegliere il grado di compressione che si desidera applicare a un'immagine in formato JPEG, ma in questo modo si determina anche la qualità dell'immagine. Più si comprime un'immagine con la compressione JPEG, più si riduce la qualità dell'immagine stessa.

Formato PNG

Il formato PNG (Portable Network Graphic) è stato sviluppato appositamente per il Web.

Questo formato è stato disponibile fin dal 1995 ma ha stentato ad acquisire popolarità a causa della mancanza di un supporto generalizzato da parte dei browser. Si tratta di un formato che secondo le intenzioni degli autori doveva sostituire il formato GIF. Questo formato senza perdita di informazioni comprime le immagini a 8 bit producendo file di dimensioni inferiori rispetto a GIF ma supporta anche immagini a 16 e 24 bit.

Anche se il formato PNG supporta il colore a 24 bit, la sua routine di compressione senza perdita di informazioni non è in grado di raggiungere l'efficienza del formato JPEG. Il formato PNG supporta, inoltre, le funzionalità di trasparenza e interallacciamento ma non l'animazione.

Un'utile caratteristica del formato PNG è la capacità di incorporare del testo per offrire la possibilità di eseguire ricerche sulle immagini; è infatti possibile memorizzare nel file dell'immagine una stringa che identifica l'immagine stessa. Purtroppo il formato grafico PNG non è ampiamente supportato e l'implementazione corrente delle immagini PNG in Netscape Navigator e Microsoft Internet Explorer non supporta completamente tutte le sue funzioni.

Compressione audio

Come nel caso delle immagini si tratta di distinguere tecniche di compressione senza perdita (lossless) e con perdita (lossy).

Compressione senza perdita

Usando un algoritmo di compressione senza perdita, dal risultato della compressione si può riottenere tutta l'informazione originaria. In questo caso la riduzione massima generalmente ottenibile, utilizzando algoritmi studiati appositamente per l'audio è all'incirca del 60%, ma solo con alcuni tipi di suono. Si possono utilizzare gli stessi algoritmi generali di

compressione (come LZW) ma i risultati in termine di riduzione sono inferiori.

Un buon esempio di algoritmo di compressione lossless è il FLAC (Free Lossless Audio Codec). FLAC è un diffuso codec audio libero di tipo lossless, cioè senza perdita di qualità. La compressione non rimuove informazioni dal flusso audio, ed è quindi adatto sia all'ascolto normale che per l'archiviazione. Il formato FLAC attualmente ha un buon supporto da parte di vari software audio.

Altri esempi di compressione lossless sono APE, ALF.

Compressione con perdita

Gli studi di psicoacustica hanno permesso di accertare che l'uomo non è sensibile nello stesso modo a tutte le frequenze e che un suono ad alta intensità ne maschera uno con frequenza vicina ma intensità più bassa. Sfruttando queste ed altre considerazioni, si può pensare di eliminare l'informazione che non verrebbe comunque percepita ed ottenere quindi un buon rapporto di compressione.

In questo modo sono stati sviluppati algoritmi di compressioni specifici per l'audio ad altissima efficienza che sono in grado di ottenere riduzione della lunghezza dei file dell'ordine di 10 a 1 praticamente senza perdita di qualità.

Facciamo alcuni esempi.

MP3

MP3 (o, più esattamente "MPEG-1/2 Audio Layer 3") è un algoritmo di compressione audio in grado di ridurre drasticamente la quantità di dati richiesti per riprodurre un suono, rimanendo comunque una riproduzione fedele del file originale non compresso. È lo standard de facto della compressione audio.

La qualità di un file MP3 dipende dalla qualità della codifica e dalla difficoltà con il quale il segnale deve essere codificato. Buoni codificatori hanno una qualità accettabili da 128 a 160 kbit/s, la chiarezza perfetta di un brano si ottiene da 160 a 192 kbit/s. Un codificatore che ha bassa qualità lo si riconosce ascoltando persino un brano a 320 kbit/s. Per questo non ha senso parlare qualità di ascolto di un brano di 128 kbit/s o 192 kbit/s. Una buona codifica MP3 a 128 kbit/s prodotta da un buon codificatore produce un suono migliore di un file MP3 a 192 kbit/s codificato con uno scarso codificatore.

Oggi esistono buone alternative a MP3 anche se molto meno diffuse.

Vorbis

VORBIS è un algoritmo di compressione orientato alla compressione mono, stereo o 5.1 surround di segnale audio PCM campionato a 44.1 kHz o 48

kHz, con una profondità di campionamento di 16 bit o 32 bit. È comunque in grado di gestire anche segnali in ingresso differenti da quelli raccomandati.

Nel trattamento di segnale stereo musicale Vorbis ha il suo bit rate ideale intorno ai 128 kbit/s, risultando estremamente difficoltoso da distinguere rispetto all'originale in un ascolto cieco già da 192 kbit/s.

Trattandosi di un algoritmo di compressione lossy, cioè a perdita di informazioni, è l'encoder a svolgere il compito più delicato in assoluto, dovendo scegliere *quale* parte di informazione acustica sacrificare.

Vorbis è un algoritmo dall'approccio pesantemente VBR, ovvero a bit rate estremamente variabile in base al tipo di segnale sonoro che è chiamato a codificare. Per questo motivo al posto di riferirsi al valore di kbit/s Xiph.Org raccomanda di usare la nomenclatura *q*, ovvero il livello di qualità con cui è stato eseguita la codifica.

Tra i pregi che comunemente si attribuiscono a Vorbis, soprattutto riferendosi all'inevitabile paragone con lo standard de-facto MP3, vanno ricordati la maggior estensione e pulizia delle alte frequenze (sopra i 16 kHz), il supporto multicanale a livello nativo e in generale una migliore conservazione delle microinformazioni di spazialità sonora del segnale originario.

Tra i difetti attribuiti vanno citati la relativa pesantezza dell'algoritmo di decodifica rispetto al collaudato MP3, e soprattutto una certa tendenza al pre-echo, ovvero un'innaturale *fantasma sonoro* che sembra precedere di alcuni brevi istanti ogni brusco aumento di pressione sonora. L'esempio che tipicamente viene portato è quello di una sonata di pianoforte con attacchi di *fortissimo* dal silenzio, oppure il suono delle nacchere.

Advanced Audio Coding

Il formato Advanced Audio Coding (AAC) è un formato di compressione audio creato dal consorzio MPEG e incluso ufficialmente nell'MPEG-4. L'AAC fornisce una qualità audio superiore al formato MP3 con una codifica più compatta. Attualmente viene utilizzato principalmente da Apple nei suoi prodotti dedicati all'audio, difatti Apple usa una variante dell'AAC che gestisce i diritti d'autore per vendere musica attraverso il proprio negozio di musica on-line iTunes Music Store. Una compressione a 128Kbps, lo standard di iTunes Music Store corrisponde a circa 160 kbps di un mp3 a bitrate variabile.

Vari studi ed esperimenti, hanno mostrato come un AAC a 128Kbps, a differenza del formato mp3, sia pressoché identica a quella del cd originale.

Altri algoritmi di compressione lossy sono: Windows Media Audio (WMA), molto diffuso sui sistemi Windows; Dolby Digital (AC3) che può comprimere

fino a 6 canali audio, di cui 5 a piena larghezza di banda ed uno per gli effetti a bassa frequenza (LFE), fino a 384 kbit/s. Viene utilizzato nei DVD e nel sistema americano ATSC DTV, MPC Musepack è un formato opensource con una qualità maggiore dell'mp3 a parità di bitrate a scapito però delle dimensioni finali del file.

Nota sul bitrate

I file multimediali sono per loro natura connessi al tempo che scorre. In altri termini ad ogni secondo è associato un certo contenuto informativo e quindi una certa sottosequenza di cifre binarie. Il numero di cifre binarie che compongono queste sottosequenze è detto bitrate. In altre parole il bitrate è il numero di cifre binarie impiegate per immagazzinare un secondo di informazione. Questo può essere costante per tutta la durata del file o variare all'interno di esso. Ad esempio i cd musicali vengono campionati (registrati) ad una frequenza pari a 44.100Hz. Da ciò si evince che ogni secondo si hanno 44.100 valori registrati dall'ipotetico microfono che vanno poi moltiplicati per i 2 canali del suono stereo che vanno a loro volta moltiplicati per 2 poiché la registrazione avviene a 16 bit (pari appunto a 2 byte). Quindi avremo:

$$44.100 \times 2 \times 2 \times 60 \text{ (secondi)} = \sim 10 \text{ MB ogni minuto}$$

La compressione, diminuendo la lunghezza globale del file, diminuirà di conseguenza la lunghezza media delle sottosequenze ossia diminuirà il bitrate medio. Il bitrate medio diventa dunque in questi casi l'indice dell'entità della compressione. Ad esempio se il file di origine possedesse un bitrate di 1411 Kbit/s e il file compresso possedesse un bitrate medio di 320 Kbit/s, allora avremmo ridotto di un fattore pari a circa 4.5.

Compressione dei filmati

La codifica digitale e la compressione dei video è affidata ai cosiddetti codec video. Un CODEC VIDEO è un programma o un dispositivo sviluppato per descrivere un flusso video sotto forma di dati numerici adatti ad essere memorizzati su un supporto digitale. Usualmente i codec video effettuano anche una compressione dei dati in modo da ridurre l'elevata quantità di dati che compone un flusso video. La maggior parte dei codec video adottano tecniche di compressioni lossy (a perdita di informazioni) in modo da poter ridurre i dati necessari per trasmettere i flussi video anche di 20 volte o più, ma esistono anche dei codec utilizzati per applicazioni professionali che utilizzano compressioni lossless (senza perdita di informazione).

A seconda della diversa tecnica di codifica del flusso video, i codec video si dividono in due grandi famiglie: a codifica intraframe e a codifica interframe.

La codifica intraframe contraddistingue i codec che codificano e decodificano un flusso video descrivendo ogni singolo fotogramma che compone la sequenza video, rispettando quindi un approccio tradizionale alla quantizzazione video come sequenza di immagini statiche.

Nella codifica interframe invece i codec video si occupano di descrivere i cambiamenti che occorrono tra un fotogramma ed il successivo partendo da un fotogramma iniziale descritto con codifica intraframe e seguendo un approccio più innovativo alla quantizzazione video allo scopo di migliorarne l'efficienza sfruttando la capacità dei sistemi di riproduzione moderni in grado di elaborare l'informazione per poi mostrarne il risultato.

La diversa dinamica dei due approcci fa sì che la codifica intraframe è più adatta alla riproduzione di sequenze video particolarmente movimentate, descrivendo ogni singolo fotogramma infatti, un codec a codifica intraframe potrà degradare la qualità delle singole immagini all'aumentare del rapporto di compressione, ma tenderà comunque a lasciare inalterata la dinamica del movimento. I codec a codifica interframe risultano invece meno adatti alla codifica di sequenze movimentate per le quali necessitano di descrivere grossi cambiamenti tra i fotogrammi. Al contrario, in sequenze video statiche, ovvero con pochi elementi che cambiano nella scena, la codifica interframe risulta di notevole efficienza.

Ecco una breve panoramica dei principali standard:

- Il **DivX®** è un formato di compressione video sviluppato da DivX Inc. Attraverso l'apposito codec è possibile riprodurre e creare file video di questo formato. La particolarità del DivX sta nella sua versatilità nel produrre file di dimensioni ridotte di filmati di lunga durata, lasciando pressoché inalterata la qualità dell'immagine. In pratica, con le opportune impostazioni, è possibile convertire un film DVD di 6-8 Gigabyte in un file DivX di 700Mb (la dimensione di un cd rom) con una qualità video e audio più che discreta. Per questo motivo, è stato al centro di controversie per il suo utilizzo nella duplicazione e distribuzione di DVD protetti. Le prime versioni di DivX (fino alla 3) erano illegali in quanto n quanto prodotto da una copia rubata del Mpeg-4 di proprietà Microsoft. Dalla versione 4 DivX è diventato un codec indipendente dal suo predecessore ed iniziata la sua vita anche commerciale. Oggi DivX è arrivato alla versione 6.
 - H.264 Questo codec video è stato sviluppato per video ad alta qualità anche a frequenze di trasmissione dei dati inferiori rispetto alle soluzioni attuali, ed è utilizzata per qualunque tipo di periferica: dai televisori ad alta definizione HDTV e DVD, ai telefoni cellulari 3G. I servizi di broadcast basati sullo standard H.264 occupano una banda inferiore rispetto al diffuso schema di codifica MPEG-2, a una frequenza di trasmissione dei bit decisamente inferiore. Gli operatori di broadcasting possono quindi trasmettere in modo economico un
-

numero maggiore di programmi ad alta definizione. L'efficienza della compressione è migliorata di oltre il 50% rispetto al precedente MPEG-2. Attualmente i dispositivi con maggior diffusione ad utilizzare questo sistema di codifica sono l'iPod video e la console Sony PSP.

- MPEG-1 è uno standard introdotto nel 1991 da MPEG (Moving Pictures Experts Group). Originariamente è stato ottimizzato per le applicazioni video a basso *bitrate* con una risoluzione video di 352x240 pixel con 30 fotogrammi al secondo per lo standard tv NTSC oppure di 352x288 pixel con 25 fotogrammi al secondo per lo standard tv PAL. MPEG-1 non è strutturalmente limitato a questi formati in quanto può raggiungere ad esempio i 4095x4095 pixel con 60 fotogrammi al secondo, ma di fatto il sistema è stato ottimizzato per un bit rate di 1,5 Mbit/s. I Video CD utilizzano il formato MPEG-1. La qualità dell'output ai *bitrate* tipici di un Video CD è quella di un VCR.
 - MPEG-2 è un sistema di codifica digitale di immagini in movimento, che permette di comprimere i dati mantenendo una buona qualità. MPEG-2 è stato destinato al broadcast televisivo, fin dalla sua introduzione nel 1994. Una efficiente codifica per il video interlacciato e la scalabilità sono state le caratteristiche che hanno permesso di digitalizzare efficacemente i segnali televisivi. Grazie all'MPEG-2 si ottengono immagini televisive di buona qualità con bitrate compresi tra 4 e 9 Mbit/s. MPEG-2 sta anche alla base dello standard DVD.
 - MPEG-4, presentato nel 1998, è il nome dato a un'insieme di standard per la codifica dell'audio e del video digitale sviluppati dall'ISO/IEC Moving Picture Experts Group (MPEG). L'MPEG-4 è uno standard utilizzato principalmente per applicazioni come la videotelefonia e la televisione digitale, per la trasmissione di filmati via Web, e per la memorizzazione su supporti CD-ROM. MPEG-4 supporta tutte le caratteristiche degli standard MPEG-1 e MPEG-2 oltre a tutta una serie di nuove caratteristiche come la gestione tridimensionale degli oggetti (tramite un'estensione del VRML). I flussi audio e video vengono trattati dallo standard MPEG-4 come oggetti che possono essere manipolati e modificati in tempo reale. Lo standard supporta caratteristiche specificate da terze parti come una particolare gestione dei Digital Rights Management o una gestione interattiva dei contenuti.
 - REAL VIDEO è un video codec proprietario, sviluppato dalla RealNetworks. La sua prima release risale al 1997. REAL VIDEO è stato usato inizialmente per servire video streaming attraverso le reti internet ad un basso bit rate verso personal computer. Lo
-

sviluppo delle connessioni verso la banda larga ha permesso in tempi recenti di offrire filmati con una maggior qualità. Inoltre, lo streaming consente di scaricare e vedere filmati di qualsiasi tipo su apparecchi cellulari. Real Video differisce dalle codifiche normali in quanto è un formato proprietario ottimizzato esclusivamente per lo streaming attraverso il protocollo (proprietario) PNA oppure tramite il Real Time Streaming Protocol. Può essere usato per scaricare e vedere video o per uno streaming live.

- Windows Media Video (WMV) è il nome generico per una serie di tecnologie proprietarie sviluppate da Microsoft per lo streaming di file video. Fa parte della piattaforma Windows Media. A partire dalla versione 7 (WMV1), Microsoft ha usato una sua versione modificata dello standard MPEG-4.
 - **XviD** è un codec video open source aderente allo standard MPEG-4 (profilo ASP) originariamente basato su OpenDivX. Il progetto XviD è partito nel Luglio 2001, con la chiusura del progetto OpenDivX da parte di DivXNetworks Inc., società creatrice del popolare codec DivX. Il codec XviD ha avuto una considerevole diffusione soprattutto nell'ambito del file sharing, dove viene utilizzato principalmente per comprimere la parte video dei film in modo che possano occupare poco spazio ed essere trasferiti velocemente nelle reti, appunto, di file sharing. La diffusione sul p2p di questo codec e il fatto che aderisse allo standard MPEG-4 ASP hanno fatto sì che molti produttori di lettori DVD che già erano in grado di leggere flussi video compressi in DivX (questi particolari lettori DVD spesso vengono chiamati *lettori Stand Alone*) aggiungessero il supporto ad XviD. La diffusione di XviD, oggi, è paragonabile a quella del codec "rivale" DivX.
-

La programmazione

Abbiamo visto come le informazioni vengono codificate per potere essere elaborate dal computer. Ora passeremo a vedere gli strumenti che abbiamo a disposizione per programmare il computer, per fare in modo, cioè, che il computer elabori le informazioni che gli forniamo in modo utile per noi.

Prima di affrontare la programmazione in senso stretto faremo un breve panorama dello sviluppo che ha avuto negli anni la programmazione dei computer e degli strumenti che ci offre oggi.

Un po' di storia

All'inizio, negli anni '40, l'unico metodo per programmare era il **linguaggio macchina**. Il lavoro del programmatore consisteva nel settare ogni singolo bit a 1 o 0 su enormi computer che occupavano stanze intere e pesavano decine di tonnellate. I monitor non esistevano; i dati e i programmi si fornivano al computer su schede perforate e il computer mandava i risultati su telescriventi. Questo tipo di procedimento, non solo era faticoso, ma era riservato ad una cerchia ristretta di persone. Eppure in questo modo gli scienziati riuscirono in pochi mesi a completare i calcoli per costruire la prima bomba atomica, calcoli che se fatti a mano, con calcolatrici meccaniche avrebbero richiesto anni.

Negli anni 50, con il progresso tecnologico e la riduzione delle dimensioni e dei costi dei calcolatori, nacquero due importanti linguaggi di programmazione, il **FORtrAN** (FORmula trANslator), il cui utilizzo era ed è prettamente quello di svolgere in maniera automatica calcoli matematici e scientifici, e l'**ALGOL** (ALGOrithmic Language), altro linguaggio per applicazioni scientifiche sviluppato da Backus (l'inventore del FORtrAN) e da Naur, i quali oltretutto misero a punto un metodo per rappresentare le regole dei vari linguaggi di programmazione che stavano nascendo.

Nel 1960 venne presentato il **COBOL** (COmmon Business Oriented Language), ideato per applicazioni nei campi dell'amministrazione e del commercio, per l'organizzazione dei dati e la manipolazione dei file.

Nel 1964 fa la sua comparsa il **BASIC**, il linguaggio di programmazione per i principianti, che ha come caratteristica fondamentale quella di essere

molto semplice e, infatti, diventa in pochi anni uno dei linguaggi più utilizzati al mondo.

Intorno al 1970, però, Niklaus Wirth pensò bene di creare il **PASCAL** per andare incontro alle esigenze di apprendimento dei neo-programmatori, introducendo però la possibilità di creare programmi più leggeri e comprensibili di quelli sviluppati in basic. Si può affermare con certezza che Wirth ha centrato il suo obiettivo, considerando che ancora oggi il Pascal viene usato come linguaggio di apprendimento nelle scuole.

Pochi anni più tardi fa la sua comparsa il **C** (chiamato così perché il suo predecessore si chiamava B), che si distingueva dai suoi predecessori per il fatto di essere molto versatile nella rappresentazione dei dati. Il C, infatti, ha delle solide basi per quanto riguarda la strutturazione dei dati, però può apparire come un linguaggio assai povero vista la limitatezza degli strumenti a disposizione. Invece la sua forza risiede proprio in questi pochi strumenti che permettono di fare qualsiasi cosa, non a caso viene considerato "il linguaggio di più basso livello tra i linguaggi ad alto livello", per la sua potenza del tutto paragonabile al linguaggio macchina, mantenendo però sempre una buona facilità d'uso.

Ma la vera rivoluzione si è avuta nel 1983 quando Bjarne Stroustrup inventò il **C++** (o come era stato chiamato inizialmente "C con classi") che introduceva, sfruttando come base il C, la programmazione Orientata agli Oggetti (OO - Object Oriented) usando una nuova struttura, la classe. La programmazione orientata agli oggetti consente di dividere l'interfaccia dal contenuto, ottenendo in questo modo tanti "moduli" interagibili tra loro attraverso le interfacce, permettendo così al programmatore di cambiare il contenuto di una classe (se sono stati trovati errori o solo per introdurre delle ottimizzazioni) senza per questo doversi preoccupare di controllare eventuale altro codice che richiami la classe. Questo nuovo stile di programmazione ha completamente stravolto il modo di programmare precedente ad esso che si riduceva ad una programmazione procedurale che lasciava poco spazio al riutilizzo del codice.

Insomma il C++ è riuscito a creare un nuovo modo di programmare, o meglio di progettare un programma, rendendo il codice scritto più chiaro e soprattutto "riutilizzabile", ed è grazie a lui che oggi possiamo usare le finestre colorate di Windows che ci piacciono tanto.

Il modello Object Oriented ha avuto grande successo e dopo il C anche il Pascal, il Basic e altri linguaggi hanno avuto la loro versione OOP. Negli ultimi anni, infine, è nato JAVA che ha due caratteristiche principali:

- È il primo linguaggio progettato appositamente per la programmazione orientata agli oggetti (non è cioè l'adattamento di un linguaggio preesistente).
-

- Il programma ottenuto non è un programma destinato a *girare* in un ambiente specifico (Windows o Macintosh o Linux) ma gira su un computer virtuale, la Java Virtual Machine. Basterà installare la versione specifica della Java Virtual Machine per il sistema operativo specifico e lo stesso programma potrà girare in ambienti completamente diversi.

Altri linguaggi di programmazione da menzionare sono il LISP (1959), l'ADA (1970), lo SMALLTALK (1970) e il LOGO e il PROLOG, ecc .

Il panorama attuale

L'enorme diffusione dei personal computer ha influito anche sul mercato dei linguaggi di programmazione, una volta sicuramente un mercato marginale. Pensiamo solamente alla diffusione di HTML e dei linguaggi di scripting ad esso collegati e ci renderemo conto di quanta più gente rispetto a solo dieci anni fa oggi si avvicina in qualche modo al mondo della programmazione.

Da un lato i produttori di software hanno tentato di assecondare questo processo cercando di offrire prodotti sempre più semplici da usare. Sono così nate le versioni visuali di vari linguaggi di programmazione che rendono enormemente più semplice sviluppare programmi nei moderni ambienti grafici a finestre: Visual Basic, Visual C ++, Delphi (versione *visual* di Object Pascal), Kilyx (versione di Delphi per il mondo Unix), Visual Java, ecc.

Dall'altro gli strumenti per lo sviluppo di contenuti multimediali on-line e off-line (pagine web, animazioni interattive, ecc.) si sono dotati di linguaggi di programmazione sempre più potenti.

Il nostro interesse è soprattutto rivolto allo sviluppo di contenuti multimediali. Può essere, comunque interessante dare un rapido sguardo agli strumenti che abbiamo a disposizione anche per realizzare programmi orientati ad altri fini. Bisogna infatti tenere conto che i linguaggi di programmazione, per loro natura, possono fare benissimo alcune cose e male altre.

Per **Applicazioni Matematiche o di Ricerca** i linguaggi più adatti sono ancora i *vecchi* Fortran e Algol.

Per i grandi progetti software (sviluppo di sistemi operativi, applicativi complessi, applicazioni lato server) normalmente viene impiegato il C o il C++.

Per lo sviluppo di **programmi gestionali** in ambiente Windows lo strumento più utilizzato è Visual Basic prodotto da Microsoft. Un'ottima alternativa al Visual Basic è, da qualche anno Delphi un prodotto Borland basato sul Pascal che consente anche il porting dei programmi tra

l'ambiente Windows e l'ambiente Unix/Linux grazie a Kilyx versione Unix di Delphi. Negli ambienti diversi da Windows si utilizzano C/C++ e ambienti di sviluppo (i cosiddetti CASE) basati su C/C++.

Applicazioni multimediali

Dal punto vista del nostro corso meritano una particolare attenzione gli strumenti per lo sviluppo di applicazioni multimediali on line e off line.

Sviluppo di pagine web

L'attività di programmazione entra a vari livelli nello sviluppo di contenuti da distribuire via Internet.

Un primo livello di base è la costruzione delle pagine web. Il linguaggio usato è l'HTML (Hyper Text Markup Language) e descrive come una pagina web debba essere mostrata da un browser. L'HTML è un linguaggio a *marcatori*: tutte le istruzioni proprie del linguaggio sono inserite tra due segni specifici (nel caso di HTML tra "<" e ">") e costituiscono i cosiddetti tag. I browser cercano di interpretare i tag come istruzioni mentre il resto viene mostrato come testo. I tag che il browser non riesce ad interpretare vengono semplicemente ignorati.

Questo approccio estremamente flessibile ha consentita un grande sviluppo delle possibilità di HTML nelle varie versioni successive senza la necessità di preoccuparsi troppo della compatibilità con le versioni precedenti.

Specifici tag consentono poi di inserire nella pagine web vari tipi di componenti che a loro volta sono programmabili. Vediamo le più importanti:

- Il tag SCRIPT consente di inserire dei nuclei di codice che il browser è in grado di eseguire e che, ad esempio, possono servire ad aumentare il grado di interattività della pagina. I linguaggi a disposizione dello sviluppatore sono due: VBSCRIPT (che deriva dal Visual Basic e viene riconosciuto però solo da Internet Explorer) e JAVASCRIPT che deriva invece da JAVA ed è riconosciuto più o meno da tutti i browser.
 - Il tag APPLET che consente di inserire in una pagina web un programma scritto in JAVA. La condizione per cui il programma possa essere eseguito è che la Java Virtual Machine sia installata sul computer.
 - Il tag OBJECT (Internet Explorer su piattaforma Windows) che insieme al tag EMBED (altri browser e Internet Explorer su Macintosh) consentono di inserire nelle pagine web oggetti di varia natura gestiti da estensioni dei browser o (in Windows) dai oggetti gestiti direttamente dal sistema operativo (i cosiddetti ActiveX). Qui segnaliamo due oggetti/plugin che negli ultimi anno hanno avuto una grande diffusione: SHOCKWAVE FLASH e SHOCKWAVE
-

DIRECTOR entrambi orientati allo sviluppo di applicazioni multimediali interattive, entrambi dotati di un potente linguaggio di programmazione.

Un altro importante livello in cui l'attività di programmazione entra nello sviluppo di pagine web è la programmazione *lato server*.

I tipi di pagine web di cui abbiamo fin qui parlato sono pagine statiche. Pagine, cioè, che vengono inviate al browser dal server web esattamente come sono state composte. Molti siti web oggi usano, invece, pagine dinamiche, pagine, cioè, il cui contenuto viene composto dal server al momento della richiesta sulla base dei parametri che gli vengono passati dal browser e del contenuto di database che il server può consultare.

Questo tipo di programmazione, un tempo riservato agli specialisti, è oggi alla portata di qualsiasi programmatore. Gli strumenti più diffusi sono PERL e PHP per i server web che girano in ambiente Unix e Linux e ASP e la sua più recente evoluzione ASP.NET per Internet Information Server (il server web Microsoft).

Applicazioni multimediali off line

Per quanto riguarda la Creazione di Giochi gli sviluppatori tendono a usare (per ottenere la massima efficienza) strumenti il più vicino possibile al linguaggio dei processori quindi C++ e Assembler. Per giochi non troppo complessi, però, anche FLASH, DIRECTOR e JAVA possono essere buone soluzioni.

FLASH è probabilmente il prodotto più abbordabile per i neofiti. offre una potente gestione della grafica e delle animazioni ed è in grado di produrre applicazioni molto leggere che possono essere facilmente distribuiti anche su Internet. Le ultime versioni (dalla 7.0 in poi) di Flash sono dotate di linguaggio molto potente ed evoluto (ACTION SCRIPT 2.0 derivato da Javascript) che consente di utilizzare tecniche di Object Oriented Programming.

DIRECTOR è l'ideale per produrre applicazioni multimediali off line multiplatforma (Windows e Macintosh). Offre un linguaggio di programmazione molto potente ed è in grado di gestire molto bene risorse multimediali diverse (suono, video, immagini vettoriali e bitmap, filmati flash, ecc.). Nelle ultime versioni, inoltre, lo sviluppatore può scegliere se utilizzare la *vecchia* sintassi del LINGO (linguaggio di programmazione proprietario sviluppato appositamente per Director) o la sintassi di JAVASCRIPT (sintassi derivata da Java, la stessa utilizzata nella programmazione dei browser e da Action Script).

Concludiamo questa breve panoramica spendendo qualche parola su JAVA. Il punto di forza di JAVA è la portabilità. Se sviluppate correttamente le applicazioni Java possono girare su un'infinità di dispositivi per cui (in forme

LA PROGRAMMAZIONE

diverse) è stata sviluppata la Java Virtual Machine (Windows, Macintosh, Unix, Linux, telefonini, palmari, ecc.).

Algoritmi

Che cos'è un programma?

Un computer è una macchina in grado di eseguire un determinato set di istruzioni. Per poter svolgere un compito il calcolatore ha bisogno di qualcosa che gli dica passo passo cosa deve fare, o meglio, quale delle istruzioni che conosce deve eseguire in un determinato momento.

Un programma è un compito che il computer deve svolgere tradotto in una serie di istruzioni che il calcolatore è in grado di eseguire. L'elaboratore è in grado di eseguire istruzioni molto basiche o, come si dice, a basso livello. Un programma scritto direttamente nel linguaggio che il computer è in grado di capire sarebbe per noi quasi incomprensibile. Ci aiutano a scrivere programmi per noi comprensibili i linguaggi di programmazione.

Dato un problema o un compito è necessario quindi "tradurlo" in termini di istruzioni nel linguaggio che abbiamo scelto. Ma prima di iniziare a programmare è conveniente analizzare il nostro problema e individuarne la soluzione in maniera indipendente dal linguaggio che useremo per programmare. Descrivere la soluzione di un problema o lo svolgimento di un compito sotto forma di passi discreti, eseguibili e non ambigui significa costruire un ALGORITMO.

L'algoritmo

Un ALGORITMO si può definire come un *procedimento* che consente di *ottenere* un dato *risultato* eseguendo, in un determinato ordine, un insieme di *passi semplici* corrispondenti ad azioni scelte solitamente da un insieme finito.

Le proprietà fondamentali di un algoritmo sono: non ambiguità (il procedimento deve essere interpretabile in modo univoco da chi lo deve eseguire), eseguibilità (ogni istruzione dell'algoritmo deve poter essere eseguita senza ambiguità da parte di un esecutore reale o ideale), finitezza (sia il numero di istruzioni che compongono l'algoritmo, che il tempo di esecuzione devono essere finiti).

ALGORITMI

Prendiamo il noto problema del contadino che deve far attraversare il fiume ad una capra, un cavolo e un lupo, avendo a disposizione una barca in cui può trasportare solo uno dei tre alla volta. Se incustoditi, la capra mangerebbe il cavolo e il lupo mangerebbe la capra.

Ecco l'algoritmo che descrive la soluzione:

```
1. Inizio
2. Porta la capra sull'altra sponda
3. Porta il cavolo sull'altra sponda
4. Riporta la capra sulla sponda di partenza
5. Porta il lupo sull'altra sponda
6. Porta la capra sull'altra sponda
7. Fine
```

Come secondo esempio vediamo l'algoritmo che sta alla base di qualsiasi computer e che stabilisce come funziona la CPU:

```
finché non trovi un'istruzione di halt fai:
    preleva un'istruzione dalla memoria
    decodifica l'istruzione
    esegui l'istruzione
```

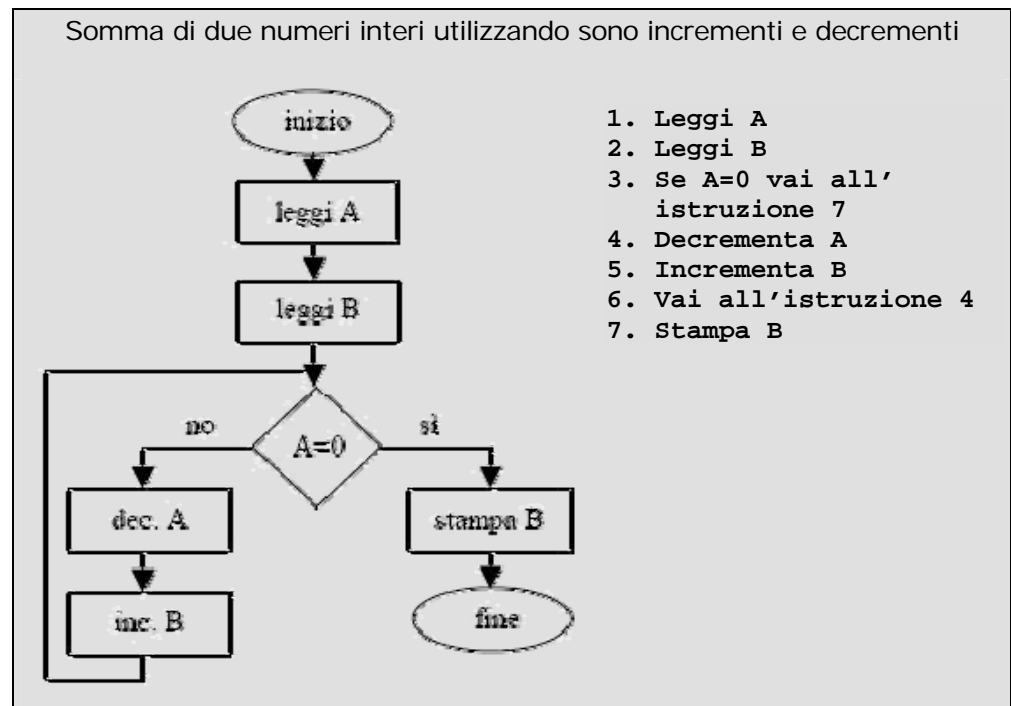
Questo algoritmo è intrinseco nella struttura della CPU e fa in modo che una volta inserito un programma nella memoria, questo possa essere letto istruzione per istruzione ed eseguito. Tale caratteristica sta alla base della programmabilità e permette la totale generalità degli elaboratori.

Rappresentazione di un algoritmo

Al fine di rappresentare gli algoritmi in modo comprensibile per il programmatore, vengono principalmente utilizzate due tecniche:

- Rappresentazione mediante pseudolinguaggio o pseudocodice che possiamo definire come un *linguaggio informale (ma non ambiguo) per la descrizione delle istruzioni*. Ogni riga rappresenta un'istruzione. Se non diversamente specificato, le righe si leggono dall'alto verso il basso.
- rappresentazione mediante diagramma di flusso (flowchart) che è una rappresentazione grafica in cui ogni istruzione è descritta all'interno di un riquadro e l'ordine di esecuzione delle istruzioni è indicato da frecce di flusso tra i riquadri.

L'esempio che segue mostra le due rappresentazioni dell'algoritmo che calcola la somma di due numeri interi usando esclusivamente istruzioni di incremento e decremento.



Definizione di uno pseudolinguaggio

Uno pseudolinguaggio si basa sul concetto di azione primitiva. L' utilizzo di azioni primitive per descrivere un procedimento fa sì che questo possa essere compreso senza alcuna ambiguità. In altre parole, nel caso in cui si descriva una azione in modo superficiale o si utilizzino vocaboli dal significato non univoco una frase potrebbe essere interpretata in diversi modi. Si prenda, per esempio, la frase *"andare a lezione può essere irritante"*. Un lettore che non conosca gli argomenti della lezione, il docente o il luogo in cui la lezione si svolge potrebbe interpretare la frase come: *"La lezione causa irritazione"* oppure *"il tragitto per arrivare dove si svolge la lezione è irritante"* oppure in entrambi i modi.

Per evitare un' interpretazione ambigua, quindi, viene definito un set di attività primitive, note agli interlocutori (uomo – macchina) e un insieme di regole per utilizzare le primitive. Tutto questo, set di primitive più l'insieme delle regole, costituisce un linguaggio di programmazione.

Ogni pseudolinguaggio deve definire un numero minimo di elementi quali:

1. Istruzioni di **start, end**
inizio e di fine

Sono i punti d' ingresso e di uscita del flusso di esecuzione

2. Istruzione di **nome←espressione**
assegnamento

L' esecuzione di un' istruzione di assegnamento avviene in due fasi: la valutazione dell' espressione (utilizzando i valori correnti delle eventuali variabili che in essa compaiono) e l' aggiornamento del valore della variabile a sinistra del segno

ALGORITMI

- | | | |
|--------------------------------------|--|---|
| 3. Scelta tra due possibili attività | if (condizione)
then (attività 1)
else (attività 2) | Biforcazione del flusso in due percorsi alternativi (mutuamente esclusivi) la cui esecuzione è condizionata al verificarsi di una condizione. L' esecuzione di un' istruzione condizionale comporta innanzitutto la valutazione della condizione, e quindi l' esecuzione delle istruzioni che compongono uno dei due cammini. |
| 4. Strutture iterative | while (condizione)
do (azione) | Esecuzione ripetuta di una o più istruzioni. La ripetizione dipende dall' esito di un controllo. È fondamentale che l' esito del controllo dipenda da variabili il cui valore può essere modificato dall'esecuzione delle istruzioni del ciclo. In caso contrario il ciclo verrebbe eseguito o mai (risultando inutile) o all' infinito (violando la definizione di algoritmo). Le variabili da cui dipende il controllo sono dette variabili di controllo del ciclo. Ogni ciclo è composto da 4 elementi: l' inizializzazione delle variabili di controllo, il controllo della condizione da cui dipende l' iterazione, il corpo del ciclo (sequenza delle istruzioni da eseguire ciclicamente) e l' aggiornamento delle variabili di controllo. |
| 5. Istruzione di salto | goto (riga di destinazione) | L'istruzione di salto sposta il flusso di esecuzione in un particolare punto dell'algoritmo. La riga di destinazione rappresenta, appunto, il punto in cui l' esecuzione riprenderà dopo il goto. |
| 6. Definizione di procedure | procedure nome (
istruzione 1
istruzione 2
istruzione 3
) | Unità di programma utilizzabili attraverso invocazione da un qualsiasi punto del codice. Tutti i linguaggi di programmazione utilizzano questa strategia per rendere più leggibile il codice. Sinonimi di procedura sono: funzione, metodo, subroutine, sottoprogramma ecc |

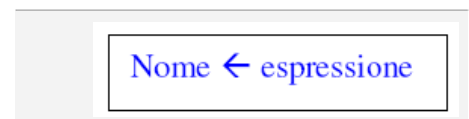
Diagramma di flusso

Il flusso inizia e termina in blocchi fittizi (detti inizio e fine) di forma generalmente arrotondata. I riquadri del diagramma di flusso possono avere forme diverse (convenzionali) per rappresentare graficamente il tipo di istruzione che contengono. Normalmente si usano solo tre tipi di riquadri: rombi con una freccia entrante e due uscenti per indicare diramazioni di flusso condizionate, rettangoli per indicare qualsiasi istruzione che non sia una condizione, ellissi per indicare inizio e fine.

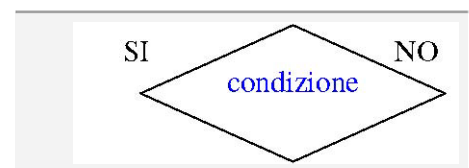
1. Inizio e fine



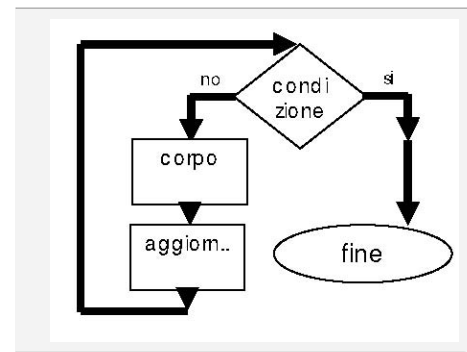
2. Assegnamento ed elaborazione



3. Scelta condizionata



4. Iterazione e salti



Esecuzione dell'algoritmo

Un algoritmo può essere visto come un risolutore di problemi che acquisisce dati d'ingresso e produce risultati. Generalmente, uno stesso algoritmo può essere applicato più volte per risolvere lo stesso tipo di problema con dati d'ingresso diversi, ovvero istanze diverse dello stesso problema. Chiamiamo esecuzione di un algoritmo la sua applicazione a determinati dati d'ingresso. Un algoritmo è detto lineare se l'esecuzione segue un flusso lineare dalla prima all'ultima istruzione, altrimenti è detto non lineare. (Es: L'algoritmo per la somma di due numeri naturali dell'esempio visto in precedenza è non lineare). Chiamiamo passo d'esecuzione l'esecuzione di un'istruzione. Il numero di passi d'esecuzione è generalmente diverso dal numero di istruzioni utilizzate per descrivere l'algoritmo e dipende dai dati d'ingresso.

Durante l'esecuzione dell'algoritmo entrano in gioco tre tipi di grandezze:

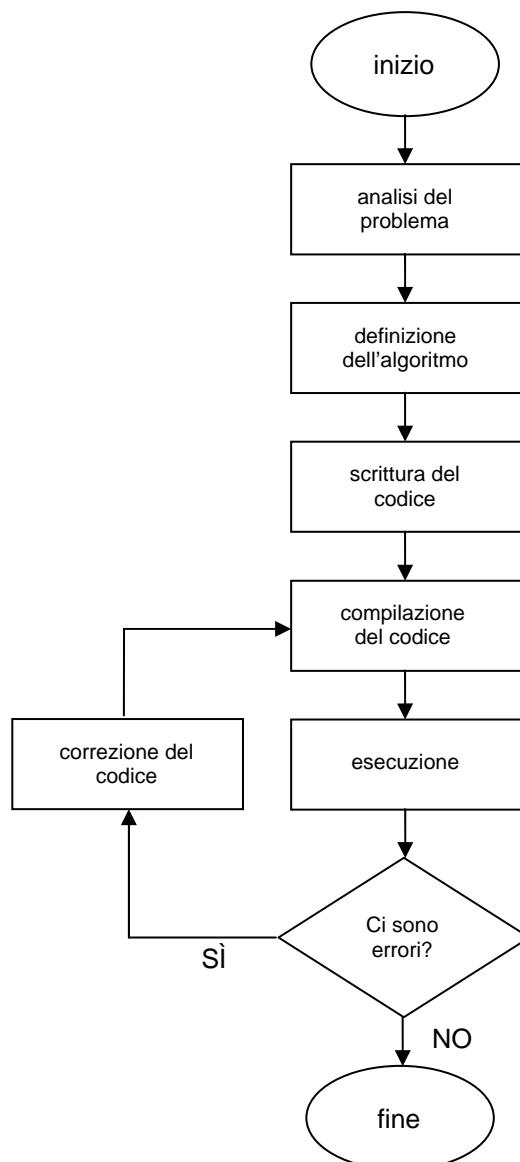
- Costanti: le costanti sono quantità note a priori il cui valore non dipende dai dati d'ingresso e non cambia durante l'esecuzione dell'algoritmo (ad esempio lo 0 dell'istruzione 4 dell'algoritmo per la somma di due numeri interi descritto precedentemente).
- Variabili: le variabili sono nomi simbolici cui sono assegnati dei valori che possono dipendere dai dati d'ingresso e cambiare durante l'esecuzione dell'algoritmo ad esempio A e B nell'algoritmo per la somma di due numeri naturali descritto precedentemente).
- Espressioni: un'espressione è sequenza di variabili, costanti o espressioni combinate tra loro mediante operatori (ad esempio $a+b*4$). Nella valutazione di un'espressione si sostituisce ad ogni variabile il suo valore corrente e si seguono le operazioni secondo regole di precedenza prestabilite (o indicate da parentesi).

Una volta costruito l'algoritmo, definita cioè in maniera chiara la strategia con cui il nostro programma affronterà il problema che deve risolvere possiamo passare alla scrittura vera e propria.

La realizzazione del programma

Qualsiasi sia il linguaggio che abbiamo scelto (anche l'ASSEMBLER) il nostro codice dovrà essere tradotto, perché possa essere eseguito dal computer, in una serie di istruzioni che la CPU è in grado di interpretare ed eseguire. Questo processo, che chiameremo compilazione, a secondo dell'ambiente e del linguaggio scelto potrà avere diverse caratteristiche ma sarà comunque un passaggio necessario del nostro processo creativo.

Se utilizziamo un diagramma di flusso per descrivere il processo di creazione di un'applicazione otteniamo qualcosa del genere:



ALGORITMI

In sintesi la creazione di un'applicazione, anche semplice, passa attraverso tre fasi: la progettazione (analisi del compito che vogliamo far svolgere all'applicazione e definizione dell'algoritmo che fa in modo che un elaboratore possa svolgere quel compito, la scrittura del codice nel linguaggio di programmazione che abbiamo scelto e il debugging (verifica del buon funzionamento dell'applicazione, correzione degli eventuali errori o variazioni che ne migliorino il funzionamento).

Impariamo ad Indentare

Scrivere programmi *sensati* e *leggibili* è difficile, ma molto importante. Un primo passo (quello che abbiamo tentato di descrivere nel capitolo precedente) è dedicare il tempo necessario alla progettazione della nostra applicazione. Questa fase ci aiuterà a chiarire la logica e la sintassi del nostro lavoro.

Il secondo passo è scrivere il programma in maniera leggibile, ma non dal punto di vista sintattico, bensì facendo attenzione a come quello che scriviamo viene presentato visivamente. Se la forma che diamo al nostro codice è piacevole e, soprattutto, chiara, sarà più facile ricordarsi che compito quel pezzo di codice aveva nell'economia del nostro programma e sarà più facile individuare errori sia sintattici che semantici. Non esiste compilatore che possa dirci gli errori "logici" che stiamo facendo.

La chiarezza della scrittura si ottiene attraverso due *tecniche* molto semplici ma estremamente utili.

La prima tecnica è l'indentazione e consiste solamente nell'inserire spazi o tabulazioni (generalmente ignorati dal compilatore) per mettere subito in luce eventuali gerarchie dei cicli o delle funzioni.

Ad esempio creando una semplice tabella in HTML possiamo scriverla in questo modo:

```
<table> <tr> <td>a</td> <td>b</td> <td>c</td> </tr> <tr> <td>
<table> <tr> <td>a1</td> </tr> <tr> <td>a2</td> </tr>
</table> </td> <td>b1</td> <td>c1</td> </tr> </table>
```

ma, francamente, non si capisce esattamente come verrà presentato il testo nel documento HTML; per questo con l'inserimento di alcuni spazi si riesce a rendere più chiaro per noi e per gli altri quello che stiamo scrivendo,

```
<table>
  <tr>
    <td>a</td>
    <td>b</td>
    <td>c</td>
  </tr>
  <tr>
```

```

<td>
<table>
  <tr>
    <td>a1</td>
  </tr>
  <tr>
    <td>a2</td>
  </tr>
</table>
</td>
<td>b1</td>
<td>c1</td>
</tr>
</table>

```

La seconda tecnica è quella di anche di "commentare" il codice scritto in maniera da rendere più chiare le operazioni logiche che stiamo svolgendo. Possiamo commentare un qualsiasi codice in svariati modi, qui sotto ne elenchiamo alcuni di esempio presi dai più comuni linguaggi di programmazione:

```
//  o  #
```

Posto dopo un'istruzione, tutto ciò che compare sulla stessa linea dopo questi simboli verrà interpretato come commento.

```
/*  */  o  <!--  -->
```

Ogni carattere compreso tra questi simboli verrà interpretato come commento

Inoltre poiché qualsiasi monitor è limitato in larghezza sarebbe buona norma, quando si scrive un programma od anche un semplice file HTML, non superare, mentre si scrive, le 80/90 colonne, poiché oltre tale limite siamo costretti ad usare la scrollbar dell'editor per continuare a leggere il testo, perdendo così del tempo e rischiando di perdere il filo logico del documento.

Introduzione alla logica

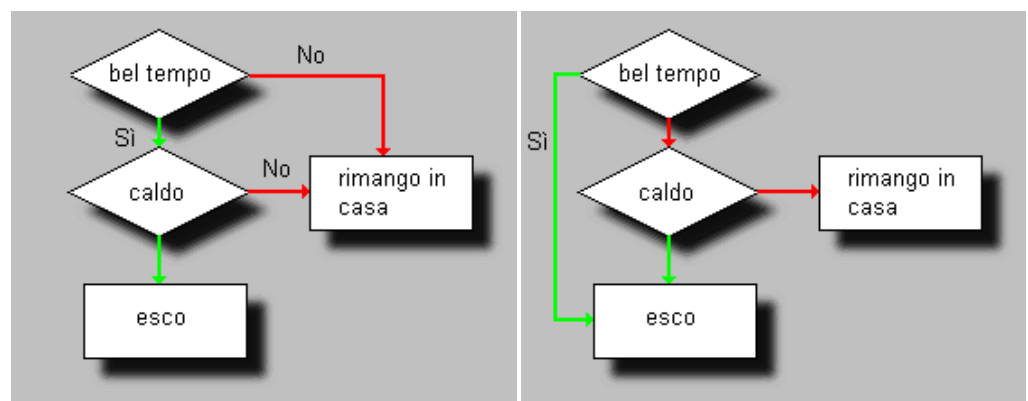
L'algebra di Boole

George Boole, nel 1854, pubblicò un libro, *An Investigations of the Laws of Thought* (Un esame sulle leggi logiche del pensiero), in cui dimostrava che la maggior parte del pensiero logico, privata di particolari irrilevanti e verbosità, potesse essere concepita come una serie di scelte. Questa idea è divenuta la base dei computer.

In questa presentazione l'algebra di Boole, piuttosto che con il suo formalismo matematico, verrà proposta in modo da renderla più simile al linguaggio naturale: in tal modo, risulterà più intuitivo comprendere il funzionamento di quei semplici circuiti digitali che costituiscono la base dei computer.

Come abbiamo visto nei capitoli precedenti qualsiasi processo logico può essere ricondotto ad una sequenza di eventi elementari, che nell'insieme prende il nome di algoritmo e tale sequenza può essere rappresentata con un diagramma di flusso (il quale a sua volta è facilmente traducibile in un particolare programma comprensibile dall'elaboratore).

Prenderemo le mosse per la nostra discussione riferendoci ad un "problema" del tutto comune. Prendiamo questi due enunciati: "esco se è bel tempo ed è caldo"; "esco se è bel tempo o se è caldo".



E' facile riconoscere che le due decisioni sono distinte: la prima (diagramma a sinistra), comporta il verificarsi di due condizioni (evidenziate in verde); la

seconda (diagramma a destra), comporta il verificarsi di almeno una fra due condizioni. E' importante rendersi conto che, come vedremo, l'elaboratore non "comprende" il significato delle frasi "esco... resto in casa": si limita a considerare il valore di certe costanti associate ad ogni singolo evento.

Come secondo passo, si tratta di convertire i diagramma di flusso in un linguaggio numerico, il solo comprensibile dall'elaboratore. Ciò si ottiene con i cosiddetti operatori logici elementari. Per semplicità, limiteremo la nostra discussione ai tre elementi di base.

Con le istruzioni riportate nella tabella a fianco, possiamo tradurre i due differenti diagrammi di flusso in sequenze di istruzioni.

operazione	istruzione	porta logica
controllo	IF (se...)	
azione	THEN (allora...)	
congiunzione	AND (...e...)	AND
separazione	OR (... oppure...)	OR
negazione	NOT (negazione)	NOT

Per far questo, è necessario aggiungere un nuovo simbolo grafico (un parallelogramma) di inizializzazione ai nostri diagrammi di flusso. Questo simbolo implica che l'elaboratore si attende che gli vengano forniti i valori di A e B (che possono essere 0 o 1) tramite tastiera. Appena inseriti questi valori (per es. $A=1$ e $B=1$), l'elaboratore esegue il programma tenendo conto dei valori di inizializzazione ($C=1$ "esco"; $C=0$ "rimango in casa"): a seconda del risultato ottenuto, sullo schermo verrà mostrata una delle due frasi.

$A = 1$ corrisponde all'evento "bel tempo"

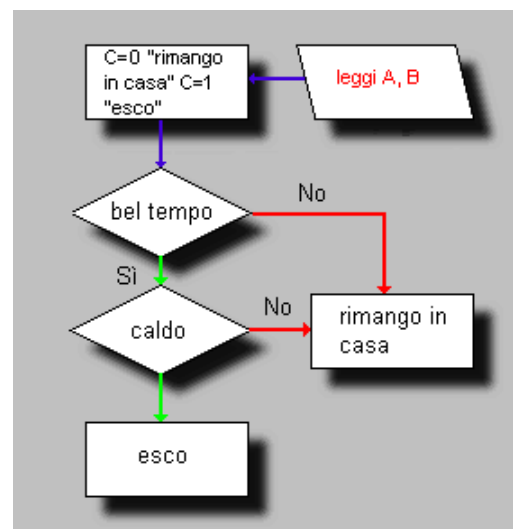
$B = 1$ corrisponde all'evento "caldo"

$C = 1$ corrisponde all'azione "esco"

$A = 0$ corrisponde all'evento "non bel tempo"

$B = 0$ corrisponde all'evento "non caldo"

$C = 0$ corrisponde all'azione "resto in casa"



con queste condizioni, il primo diagramma di flusso risulta così formalizzato:

IF A AND B THEN C

Le istruzioni **AND** e **OR**, dette operatori logici, sono prese dall'algebra di Boole e forniscono il risultato 1 o 0, a seconda del valore delle variabili A e B . Ad esempio, se entrambe le variabili A e B valgono 1, allora l'operatore AND assumerà il valore 1. In questo caso, il simbolismo, tradotto in parole, corrisponde a:

SE *bel tempo* **E** *caldo* **ALLORA** *esco*

viceversa, se una sola delle due variabili, oppure entrambe valgono 0, allora l'operatore AND assumerà il valore 0. In questo caso, il simbolismo, tradotto in parole, corrisponde a:

SE *non bel tempo* **E** *caldo* **ALLORA** *resto in casa*
SE *bel tempo* **E** *non caldo* **ALLORA** *resto in casa*
SE *non bel tempo* **E** *non caldo* **ALLORA** *resto in casa*

Come abbiamo detto, i valori delle variabili A e B sono introdotti da tastiera prima di "raggiungere" l'operatore AND. Così, se entrambe le variabili A e B valgono 1, quando saranno esaminate dall'operatore AND seguirà necessariamente il risultato $C = 1$.

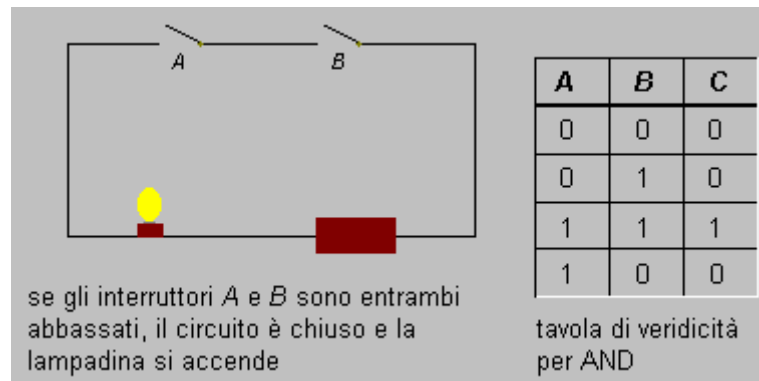
Con le stesse modalità, il secondo diagramma di flusso risulta così formalizzato:

IF A OR B THEN C

In questo caso, se almeno una delle variabili A e B vale 1, allora l'operatore OR assumerà il valore 1; viceversa, se entrambe valgono 0, allora l'operatore OR assumerà il valore 0. Dunque, se almeno una delle variabili A e B vale 1, quando saranno esaminate dall'operatore OR seguirà necessariamente il risultato $C = 1$.

Ora che abbiamo visto come definire un linguaggio numerico (ricordiamo che l'elaboratore *non* comprende il significato delle frasi che leggiamo) interpretabile dall'elaboratore, vediamo come applicarlo. Per far questo, esaminiamo un semplice "elaboratore" in grado di automatizzare la nostra decisione. La realizzazione di questo - e di tutti gli elaboratori - richiede la costruzione di circuiti detti porte logiche.

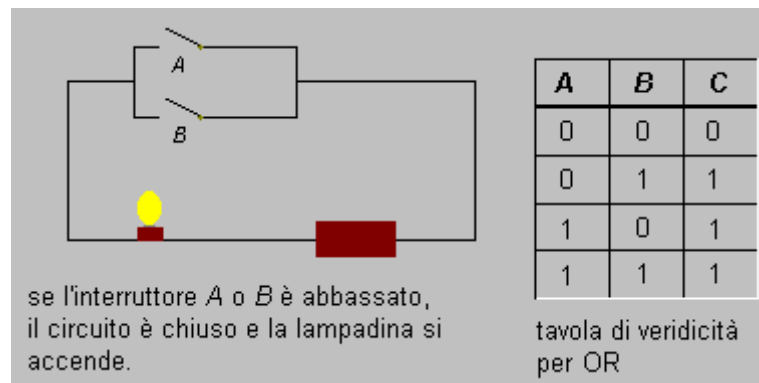
Per esempio, osservando il circuito elettrico schematizzato in figura, è facile riconoscere che la lampadina si accenderà solo se entrambi gli interruttori A



e B verranno abbassati in modo da chiudere il contatto elettrico. Questo circuito, corrisponde ad un operatore logico AND.

Per riassumere il comportamento di una porta logica, si ricorre alle cosiddette tavole di veridicità. Per la porta AND, la tavola è in tabella a destra del circuito. Si osservi, per esempio, che quando $A=1$ e $B=1$ (entrambi gli interruttori abbassati), allora la lampadina (indicata con C) è accesa e quindi $C=1$.

Ora, come secondo passo, esaminiamo un altro circuito: in questo caso, è facile riconoscere che la lampadina si accenderà solo se almeno uno degli

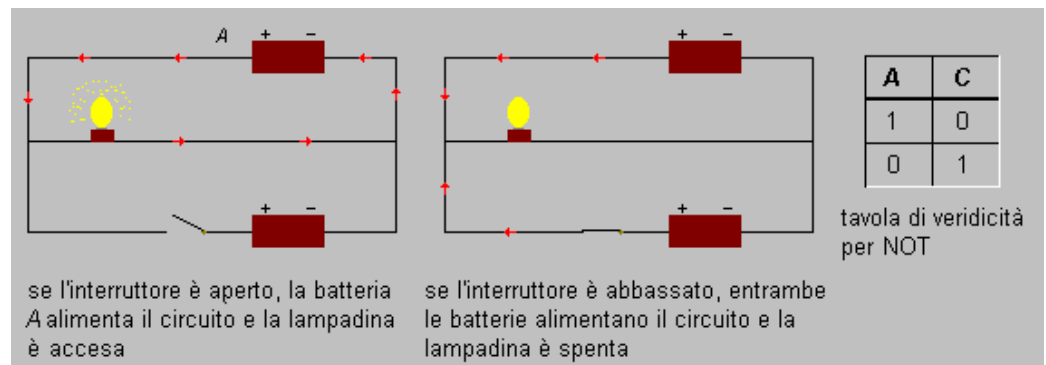


interruttori A e B verrà abbassato in modo da chiudere il contatto elettrico. Questo circuito, corrisponde all'operatore logico OR.

La sua rappresentazione secondo una normativa standardizzata, è rappresentata con il simbolo sottoindicato. A destra, è riportata la corrispondente tavola di veridicità. Si noti come sia sufficiente che un solo interruttore sia abbassato per accendere la lampadina.

Oltre alle porte AND e OR, c'è la porta NOT, capace di invertire il segnale in ingresso: se vale 1, diventa 0 e viceversa.

Il corrispondente circuito, comprende due alimentatori con polarità opposta.



Quando il circuito inferiore è aperto (0), la batteria A alimenta il circuito superiore e la lampadina è accesa; quando il circuito inferiore è chiuso, la seconda batteria fornisce una corrente uguale e opposta che impedisce il passaggio di corrente per cui la lampadina si spegne. La tavola di veridicità corrispondente a questo circuito, comprende un solo ingresso. Si osservi che quando l'interruttore è alzato ($A=0$), la lampadina è accesa ($C=1$) e viceversa.

In sintesi

Riassumiamo nelle tabelle seguenti i risultati che restituiscono gli operatori logici.

AND – Congiunzione

falso AND falso	risultato falso
falso AND vero	risultato falso
vero AND falso	risultato falso
vero AND vero	risultato vero

AND (in Action script &&) restituisce un valore vero "se e solamente se gli altri due valori sono veri", questo vuol dire che anche in una successione di più operazioni AND basta che un valore sia falso ed anche il risultato lo sarà.

OR - Disgiunzione

falso OR falso	risultato falso
falso OR vero	risultato vero
vero OR falso	risultato vero
vero OR vero	risultato vero

L'OR (in Action Script ||) restituisce un valore vero "se e solamente se almeno uno dei due valori risulta vero"; in poche parole anche in una successione di più operazioni OR basta che un valore sia vero ed anche il risultato lo sarà.

NOT – Negazione

NOT **falso**

risultato **vero**

NOT **vero**

risultato **falso**

Il NOT (in Action Script !) si riduce ad una semplice operazione di negazione del valore acquisito.

Gli elementi del linguaggio

Prendiamo ora in esame la grammatica di base di ogni linguaggio di programmazione, i mattoni con cui costruiremo i programmi.

Come punto di riferimento prenderemo il linguaggio con cui avremo a che fare in questo corso: ActionScript il linguaggio utilizzato da FLASH e che deriva da JavaScript, ma la maggior parte delle cose di cui parleremo varranno, in generale, per tutti i linguaggi.

I termini utilizzati all'interno di un linguaggio si suddividono in Parole chiave, Operatori e separatori, Letterali (o Costanti) e Identificatori definiti dal programmatore (nomi di classi, di variabili, di funzioni). Queste unità semantiche di base del linguaggio le definiremo token. Ora le vedremo in dettaglio, ma prima di iniziare è necessaria una premessa importante. Alcuni linguaggi sono case sensitive altri no. In ActionScript e Java (ma anche, ad esempio, in C e in JavaScript) le parole While, while e WHILE sono per il compilatore parole differenti e solo while è un'istruzione del linguaggio. In altri linguaggi (Pascal, Basic, ecc.) le parole hanno lo stesso significato indipendentemente dal fatto che i caratteri che le compongono siano maiuscoli o minuscoli.

È necessario, quindi, fare molta attenzione. Come vedremo, molti errori derivano, banalmente, dall'errata scrittura di parole chiave o identificatori. In ActionScript la cosa è ulteriormente complicata dal fatto che fino a Flash 6 ActionScript NON era case sensitive, dalla versione 7 di Flash è, invece, diventato case sensitive. Codice scritto per Flash 6 potrebbe quindi non girare in Flash 7 solo perché qualche parola chiave o qualche nome di variabile non è stato scritto con le maiuscole al posto giusto.

Parole chiave

Le parole chiave sono le parole, o meglio i termini composti da caratteri alfanumerici, riservate al linguaggio di programmazione. Il creatore del linguaggio di programmazione stabilisce a priori quali termini riservare e quale sarà la loro funzione, il compito del programmatore è quello di impararle ed usarle in maniera appropriata. Fortunatamente l'uso improprio

di tali termini viene generalmente rilevato durante la fase di compilazione di un programma.

Qui di seguito vengono riportate, come esempio, le parole chiave di ActionScript 2.0.

```
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
```

Ecco, invece, ad esempio, quelle del linguaggio JAVA:

```
abstract, assert, boolean, break, byte, case, catch, char,
class, const, continue, default, do, double, else, enum,
extends, final, finally, float, for, goto, if, implements,
import, instanceof, int, interface, long, native, new,
package, private, protected, public, return, short, static,
strictfp, super, switch, synchronized, this, throw, throws,
transient, try, void, volatile, while
```

Come potete vedere molto sono presenti in entrambi i linguaggi.

Le parole chiave sono parole riservate, non possono essere cioè usate per come nomi definiti dal programmatore. La ragione è piuttosto semplice da capire: se assegno ad una funzione il nome `new` o ad una variabile il nome `this`, il compilatore non sarebbe in grado di distinguere tra l'uso di quella parola come elemento fondamentale del linguaggio di programmazione o semplice nome da utilizzare per memorizzare dati.

Operatori e separatori

In combinazione con le parole chiave vengono usati alcuni token composti di uno o più caratteri speciali che servono a controllare il flusso delle operazioni che dobbiamo eseguire o a costruire espressioni, questi token vengono chiamati **operatori**. Questi sono gli operatori usati sia da ActionScript che da JAVA:

```
++ ! != !== % %= & && &= ( ) - * *= , . ?: / // /*
/= [ ] ^ ^= { } | || |= ~ + += < << <=< <> = -=
== === > >= >> >=> >>> >>=>
```

I **separatori** invece sono simboli di interpunzione che permettono di chiudere un'istruzione o di raggruppare degli elementi. Il separatore principale è lo spazio che separa i token tra di loro quando non ci sono altri separatori. Gli altri separatori sono:

```
( ) { } , ;
```

Alcuni simboli possono essere usati indistintamente come separatori o come operatori (ad esempio la parentesi), l'utilizzo nella maniera più appropriata si evince dal contesto.

Gli operatori hanno delle proprietà che ne caratterizzano l'uso semplice o composito, tali proprietà permettono infatti di interpretare in modo unico il significato di un'espressione.

Ecco le proprietà degli operatori:

- **POSIZIONE**

La posizione di un operatore rispetto ai suoi operandi (detti anche argomenti) è molto importante. Un operatore si dice **prefisso** se viene posto prima degli operandi, **postfisso** se viene posto dopo e **infixo** se si trova tra gli operandi.

- **ARIETÀ**

L'arietà è il numero di argomenti di un operatore; ad esempio il simbolo "+" (che incrementa di uno) ha un solo argomento, il simbolo "+" e le parentesi hanno, invece, due argomenti, l'operatore "?:" (operatore condizionale, simile all'IF) è l'unico ad avere tre argomenti.

- **PRECEDENZA (O PRIORITÀ)**

Per precedenza si intende l'ordine con il quale verranno eseguiti gli operatori. L'esempio classico è quello dell'espressione **4+7*5** in cui non sappiamo se bisogna eseguire prima l'operazione di somma o di moltiplicazione; premesso che l'operatore di moltiplicazione ha una priorità maggiore rispetto all'addizione, il modo corretto di interpretare l'espressione sarà **4+(7*5)**.

- **ASSOCITIVITÀ**

Questa proprietà ci viene in aiuto quando eseguiamo due o più operatori aventi stessa precedenza. Un operatore può essere associativo a **sinistra** oppure associativo a **destra**, questo vuol dire che iniziamo ad eseguire gli operatori partendo rispettivamente da sinistra o da destra.

Letterali (o costanti)

Le costanti (o letterali) sono quantità note a priori il cui valore non dipende dai dati d'ingresso e non cambia durante l'esecuzione del programma. Le costanti possono essere numeri, stringhe di caratteri, valori di verità.

Costanti numeriche

I token che indicano costanti numeriche iniziano sempre con un carattere numerico: il fatto che un token inizi con un numero basterà ad indicare al

compilatore che si tratta di una costante numerica. Se il compilatore non potrà valutare quel token come numero segnerà un errore.

Il segno che separa la parte intera di un numero dalla parte decimale è il punto.

È possibile inserire numeri in formato decimale, binario, ottale o esadecimale. Per scrivere una costante numerica in formato esadecimale oltre ai numeri e al punto dovremo usare anche delle prime sei lettere dell'alfabeto. Per segnalare al compilatore che un numero non è decimale si fa precedere il numero da un prefisso. Per i numeri esadecimali (gli unici che ci capiterà di usare) questo prefisso è "0x" (che non a caso inizia con un numero, il numero zero).

Un'ultima nota. Da quanto abbiamo appena detto deriva che tutti gli altri token (parole chiave e nomi) NON possono iniziare con un numero.

Alcuni esempi di costanti numeriche:

```
1
2433
1000000000
3.14
.3333333333
0.5
2345.675
0xFF0088
0x5500ff
0xff.00aa
```

Costanti stringa

Una stringa è una sequenza di caratteri UNICODE ed permette di rappresentare testi. Un costante stringa è una sequenza (anche vuota) di caratteri racchiusi tra apici singoli o apici doppi.

Racchiudendo la sequenza tra apici singoli occorre prestare attenzione all'uso degli apostrofi. Nel set di caratteri UNICODE, l'apostrofo e l'apice singolo coincidono. Vedremo più avanti quali problemi possono sorgere e come evitarli.

Per inserire ritorni a capo, tabulazioni, particolari caratteri o informazioni di formattazione si utilizzano speciali sequenze di caratteri dette sequenze di escape. Una sequenza di escape è formata da un carattere preceduto dal simbolo "\" (backslash). La sequenza di escape inserisce un carattere che non sarebbe altrimenti rappresentabile in un letterale stringa.

Le principali sequenze di escape sono:

```
\n: nuova riga;
\r: ritorno a capo;
```

```
\t: tabulazione orizzontale;
\' : apostrofo (o apice singolo);
\" : doppio apice;
\\ : backslash(essendo un carattere speciale deve essere
inserito con una sequenza di escape).
```

Ecco alcuni esempi di letterali stringa corretti:

```
// Stringa racchiusa da apici singoli
'Ciao a tutti'
// Stringa racchiusa tra apici doppi
"Ciao"
/* La sequenza di escape risolve l'ambiguità tra l'apostrofo
inserito nella stringa e gli apici singoli che la
racchiudono */
'Questo è l\'esempio corretto'
/* In questo caso non c'è ambiguità perché la stringa è
racchiusa tra doppi apici */
"Anche questo è l'esempio corretto"
/* Per inserire un ritorno a capo si usano le sequenze
di escape */
"Questa è una stringa valida\rdi due righe"
```

I seguenti sono invece esempi di letterali stringa non corretti:

```
/* Nel caso seguente l'apostrofo è interpretato come
l'apice di chiusura della stringa */
'Questo è l'esempio errato'
/* Il tipo di apici che inizia la stringa e quello che
la chiude devono essere uguali */
"Questa stringa non è valida"
// I letterali stringa devono comparire su un'unica riga
"Anche questa stringa
non è valida"
```

Costanti booleane

Le costanti booleane, poiché rappresenta valori logici, possono avere solo due valori: vero (rappresentato dal letterale true) e falso (rappresentato dal letterale false).

Array e oggetti

In ActionScript e in altri linguaggi è possibile inserire anche costanti di tipo Array e costanti di tipo Object. Chiariremo meglio in seguito cosa sono array e oggetti. Per ora basti dire che entrambi consentono di organizzare più valori nello stesso contenitore: l'array è un elenco di elementi ai cui valori si può accedere per indice (che è il numero che rappresenta la posizione dell'elemento nell'array) mentre gli oggetti rappresentano un gruppo di proprietà il cui valore è associato a una chiave (il nome della proprietà).

Il letterale Array è costituito da una serie di elementi separati da virgole compresa tra due parentesi quadre:

```
// array che contiene i mesi dell'anno
["January", "February", "March", "April"];
```

Il letterale Object è invece compreso tra parentesi graffe ed è costituito da una serie di copie "chiave:valore" separate da virgole:

```
//record di una rubrica telefonica in formato Object
{name:"Irving",age:32,phone:"555-1234"};
```

Altre costanti

I linguaggi normalmente definiscono anche altre costanti. Alcune rappresentano valori speciali che non possono essere rappresentati che da un valore simbolico (come abbiamo visto per true e false) altre sono valori rappresentati da un letterale per comodità, ma possono essere anche rappresentate, a seconda dei casi, come stringe o come valori numerici.

Per quanto riguarda il primo gruppo ActionScript definisce Infinity, -Infinity, undefined, null e NaN: Infinity e -Infinity rappresentano l'infinito positivo e l'infinito negativo, undefined è il valore speciale che contiene una variabile prima di essere usata, null è un valore che indica che una variabile non contiene nulla (vedremo in pratica la differenza tra questi due valori), NaN (significa Not a Number, attenzione all'uso corretto di maiuscole e minuscole!) è il valore che restituisce un'espressione numerica quando il calcolo non può essere eseguito (normalmente perché uno degli operandi non è un numero). NaN, null e undefined sono definiti da molti altri linguaggi.

Il secondo gruppo è molto vario e cambia molto da linguaggio a linguaggio. ActionScript definisce newline che equivale a "\r" (ritorno a capo) come costante globale. Molte altre costanti vengono definite come proprietà di classi predefinite. La classe Math, ad esempio, fornisce una serie di utili costanti numeriche: Math.PI (3.14159265358979 o π), Math.SQRT2 (1.4142135623731, radice quadrata di due), ecc.

Identificatori

Un identificatore è un nome definito dal programmatore. Gli identificatori si usano per dare nomi alle variabili, alle funzioni e ai tipi di dati (o classi). Parleremo in dettaglio di questi elementi nei capitoli seguenti. Qui basterà ricordare le regole a cui attenersi nella scelta degli identificatori:

- il primo carattere deve essere una lettera o il simbolo "_" (ricordiamo che nel caso la prima lettera fosse un numero il compilatore tenterebbe di interpretare il token come costante numerica);
 - i caratteri successivi possono essere lettere, numeri o "_".
-

GLI ELEMENTI DEL LINGUAGGIO

Gli identificatori non possono inoltre coincidere con le parole riservate del linguaggio.

Variabili e Tipi di Dati

Variabili

Adesso è fondamentale definire il concetto di variabile, visto che nelle prossime lezioni faremo largo uso di questo termine.

Pensiamo, ad esempio, a quando salviamo un numero di telefono del nostro amico Mario sul cellulare; se vogliamo chiamare il nostro amico, basterà inserire il suo nome (Mario, nome della variabile) ed il cellulare comporrà automaticamente il numero di telefono (valore della variabile). Se per qualche ragione Mario cambierà numero di telefono, modificherò il contenuto della mia rubrica (cambierò il valore della variabile). In questa maniera senza modificare le mie abitudini (inserirò sempre Mario) il mio cellulare comporrà il nuovo numero.

Si può vedere quindi che una variabile è composta da due elementi: il suo nome e il suo valore; come ho visto nell'esempio del cellulare in un programma posso usare i nomi delle variabili al posto dei valori che rappresentano. Questo ha due vantaggi. Il primo è la possibilità di usare simboli mnemonici al posto di numeri e stringhe di grande entità o difficili da ricordare (mi è più comodi scrivere Mario che il suo numero di telefono che potrei non ricordare). Il secondo è la possibilità di usare il nome della variabile al posto del suo valore per eseguirvi sopra delle operazioni, con la possibilità, in seguito, di modificare il valore come e quante volte vogliamo, in altri termini di generalizzare un'elaborazione.

Facciamo un esempio. Proviamo a scrivere un programma che calcola il quadrato di un numero inserito dall'utente e lo mostra sul video. Per il momento non useremo un linguaggio specifico. Useremo uno pseudolinguaggio inventato da noi. Come vedete "A" può assumere un valore a piacere. In questa maniera possiamo generalizzare l'operazione.

```
scrivi sullo schermo "Ciao Inserisci un numero";  
A = -numero inserito da tastiera-;  
B = A * A;  
scrivi sullo schermo "Il quadrato di " A " è " B;  
/* Anche B è una variabile e viene usata per registrare il  
risultato finale */
```


Se inseriamo come numero il valore "4", sullo schermo verrà scritto "Il quadrato di 4 è 16", se scriviamo 2 sullo schermo verrà scritto "Il quadrato di 2 è 8" e così via...

Tipi

Le variabili possono contenere vari tipi di dati. Un tipo di dato o, più semplicemente un tipo definisce come le informazioni verranno codificate per essere elaborate o semplicemente memorizzate.

Prima di usare una variabile è necessario dichiararla. La dichiarazione è un comando che comunica al compilatore che un determinato letterale è il nome di una variabile e che quella variabile conterrà un determinato tipo di dati. La dichiarazione consente al compilatore di controllare alcuni errori e di segnalarceli: ad esempio se sbagliamo a scrivere il nome di una variabile o se assegniamo alla variabile un valore di un tipo diverso da quello che avevamo dichiarato il compilatore ce lo segnala, facilitandoci la fase di debugging.

In linea di massima esistono due categorie di tipi di dati. I tipi primitivi e i tipi derivati.

Tipi primitivi

I tipi primitivi sono quelli fissati dalle specifiche del linguaggio.

La prima caratteristica dei tipi primitivi è che possiamo manipolarli utilizzando degli operatori.

Quando dobbiamo operare su un dato, lo dobbiamo mettere in una variabile. Una variabile è come una scatola che ci consente di maneggiare e di memorizzare i dati. Un'altra caratteristica dei tipi primitivi è che la scatola li contiene completamente.

Sembra ovvio, ma, come vedremo, non lo è affatto. Se pensiamo alle variabili come scatole possiamo dire che se contengono tipi primitivi aprendole troveremmo direttamente i dati, se invece contengono dati di un tipo derivato aprendole troveremmo il numero di scaffale che i dati occupano nella memoria del computer. Nel primo caso la variabile conterrà direttamente il dato. Nel secondo caso conterrà il puntatore al dato ossia l'indirizzo di memoria in cui il dato è stato collocato. Come vedremo in seguito questa differenza non è affatto secondaria.

I tipi primitivi definiti in ActionScript sono :

Boolean

Il tipo di dati Boolean può avere due valori: true e false. Nessun altro valore è consentito per le variabili di questo tipo. Il valore predefinito di una variabile booleana dichiarata ma non inizializzata è false.

VARIABILI E TIPI DI DATI

Number

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 e quindi i risultati delle operazioni vengono arrotondate al numero intero più vicino.

String

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.

Ecco alcuni esempi di dichiarazione di variabili di tipi primitivi:

```
//dichiarazioni di variabili in actionscript
/* a può contenere solo un numero, s una stringa k true o
   false */
var a:Number;
var s:String;
var k:Boolean;

/* per b e messaggio oltre a dichiarare il tipo viene
   Impostato un valore iniziale */
var b:Number = 1;
var messaggio:String = "Ciao a tutti";
```

Come potete vedere dagli esempi all'atto della dichiarazione si può assegnare un valore alla variabile (inizializzazione).

Finché alla variabile non viene assegnato un valore essa conterrà il valore speciale undefined.

Tipi derivati o complessi

I tipi primitivi servono a rappresentare dati semplici: un numero, vero o falso, una stringa di caratteri. Per rappresentare dati più complessi (ad esempio un elenco di valori, i dati che compongono un indirizzo, una data, ecc.) ho a disposizione alcuni tipi complessi che il linguaggio mi offre oppure ne posso creare ad hoc.

Come anticipavamo la differenza fondamentale tra i tipi primitivi e i tipi derivati è che nel primo caso le variabili contengono direttamente i dati (vero, falso, un numero, una stringa) nel secondo caso la variabile contiene il *puntatore* cioè il numero della casella, l'indirizzo in cui il dato è memorizzato sul computer.

La programmazione Condizionale

Il modo in cui strutturiamo un ragionamento, generalmente, segue la logica procedurale, cioè avendo delle operazioni da fare le eseguiamo una dopo l'altra in sequenza dalla prima all'ultima; anche nella programmazione avviene la stessa cosa, infatti quando scriviamo un programma dobbiamo pensare a creare una schermata introduttiva, chiedere i dati all'utente e poi fornirgli il risultato. Senza entrare nel dettaglio, perché questa guida è per principianti cercheremo ora di spiegare i costrutti più usati nei linguaggi di programmazione per "incanalare" il flusso del nostro programma verso la soluzione (vedere la lezione che parla dei Diagrammi di Flusso); questi strumenti sono chiamati Istruzioni Condizionali perchè permettono di eseguire del codice a seconda che una condizione sia vera o falsa.

Il costrutto principale, usato universalmente, è l'**if**, il quale permette di porre una condizione ed eseguire delle scelte in base ai dati fornitigli. Qui di seguito cerchiamo di spiegarlo sempre utilizzando uno pseudolinguaggio:

```
istruzione-if :  
    if ( espressione ) istruzione  
    if ( espressione ) istruzione else istruzione
```

L'espressione che compare dopo la parola chiave **if** deve essere di tipo logico, se la condizione risulta vera viene eseguita l'istruzione subito seguente; nel secondo caso, invece, se la condizione risulta falsa si esegue l'istruzione seguente, altrimenti si esegue l'istruzione subito dopo la parola chiave **else**. Poiché le istruzioni all'interno dell'**if** possono essere anche più di una è opportuno, per non creare confusione, inserirle all'interno delle parentesi graffe '{' e '}', anche se generalmente un else si riferisce sempre all'if che gli sta più vicino. Per più scelte invece si può usare l'**else if** che permette di porre una condizione anche per le alternative, lasciando ovviamente la possibilità di mettere l'**else** (senza condizioni) in posizione finale.

Qui riportiamo un semplice programma, sempre scritto in pseudocodice, che utilizza il costrutto if e le sue varianti finora esposte:

```

intero A = 50;
scrivi sullo schermo "Inserisci un numero";
intero B = -numero inserito da tastiera-;
if (B minore di A) {
    scrivi sullo schermo "Il numero inserito è minore di
    cinquanta";
} else if (B maggiore di A) {
    scrivi sullo schermo "Il numero inserito è maggiore di
    cinquanta";
} else if (B uguale a A) {
    scrivi sullo schermo "Il numero inserito è cinquanta";
} else {
    //poiché B non è minore, né maggiore né uguale a A
    // A non è un numero
    scrivi sullo schermo "Inserisci un numero";
}

```


Questo programma aspetta che l'utente inserisca un numero tramite la tastiera (che viene memorizzato nella variabile B) e poi lo confronta con il valore predefinito (50 che viene memorizzato nella variabile A), a seconda che il valore inserito sia più grande, più piccolo o uguale a tale numero, il programma visualizza sullo schermo una frase che ci dice il risultato di tale confronto.

Esempio in Action Script

Per realizzare questo programma in in Flash con Action Script come prima cosa dovremo creare sullo stage tre campi di testo. Per fare questo utilizzeremo l'apposito tool presente nella *palette strumenti* e utilizzando la *palette proprietà* definiremo le caratteristiche dei campi.

Il primo campo sarà di tipo *testo statico* e qui inseriremo il testo *"Inserisci un numero"*. Il secondo sarà di tipo *testo di input* e servirà a raccogliere il numero inserito dall'utente: come font del campo sceglieremo *_sans* e gli daremo il nome *input_txt*. Il terzo sarà un campo di tipo *testo dinamico* e servirà a mostrare sullo schermo la frase che ci segnalerà il risultato del nostro confronto: come font del campo sceglieremo *_sans* e gli daremo il nome *messaggio_txt*.

A questo punto premendo F5 aggiungeremo un fotogramma al nostro filmato in modo che sia composto da almeno due fotogrammi e aggiungeremo un livello che chiameremo *Azioni*.

Selezioniamo il primo frame sul livello azioni e apriamo la finestra delle azione cliccando sulla palette delle proprietà il simbolo . Sulla barra della finestra delle azione dovrà essere scritto: *Azioni - Fotogramma* ad indicare che stiamo modificando le azioni per quel fotogramma.

Inseriamo il codice seguente:

```
var A:Number = 50;
var B:Number = input_txt.text;
if (B < A) {
    messaggio_txt.text = "Il numero inserito è minore di
cinquanta";
} else if (B > A) {
    messaggio_txt.text = "Il numero inserito è maggiore di
cinquanta";
} else if (B == A) {
    messaggio_txt.text = "Il numero inserito è cinquanta";
} else { //B non è un numero
    messaggio_txt.text = "Inserisci un numero!!!";
}
```

Scegliamo sul menu *controllo* e quindi *prova filmato*. È possibile trovare il filmato che contiene questo codice nella cartella Esempi del CD:

ProgrammazioneCondizionale fla.

NB: Il filmato deve essere composto di almeno due fotogrammi perché solo in questo caso gira indefinitamente in *loop*. Ogni volta che si torna al primo fotogramma, il codice che controlla quanto inserito dall'utente viene ripetuto e il contenuto dei campi aggiornato.

La programmazione Iterativa

Fino ad ora abbiamo visto che la modalità generale di programmazione è quella procedurale: viene eseguita un'istruzione dopo l'altra fino a che non si incontra l'istruzione di fine programma. Esiste anche un'altra logica, quella iterativa; un'istruzione (o una serie di istruzioni) vengono eseguite continuamente, fino a quando non sopraggiungono delle condizioni che fanno terminare tale computazione. Facciamo l'esempio più semplice, voler stampare a video per mille volte la solita frase; la programmazione procedurale suggerisce di scrivere mille volte il codice, mentre quella iterativa permette di scrivere il comando **una ed una sola volta** e poi ripeterlo per mille volte, dopo le quali una struttura di controllo adeguata (ad esempio un contatore da uno a mille) termine l'esecuzione del ciclo.

While, do e for

I tre costrutti comunemente usati sono il WHILE, il DO ed il FOR che fanno praticamente la stessa con modalità leggermente diverse.

L'istruzione WHILE viene schematizzata come segue:

```
istruzione-while :  
    while ( condizione ) blocco istruzioni
```

Con questa istruzione viene prima valutata la condizione racchiusa tra le parentesi tonde, se l'espressione risulta vera viene eseguita l'istruzione all'interno del while e il while viene ripetuto, altrimenti si esce dal ciclo del while e si procede con il resto del programma.

Il DO può essere considerato una variante dell'istruzione while ed è strutturato nella maniera seguente:

```
istruzione-do :  
    do blocco istruzioni while ( condizione )
```

Prima di tutto viene eseguita l'istruzione racchiusa tra **DO** e **WHILE** (quindi si esegue almeno una volta), poi si verifica il risultato dell'espressione, se è vero si riesegue il DO, altrimenti si continua con l'esecuzione del resto del programma. L'unica differenza rispetto al costrutto precedente è che il blocco di istruzioni da iterare viene eseguito almeno una volta.

Il FOR, schematizzato qui sotto, inizializza una variabile, pone una condizione (che deve essere vera o falsa) e poi modifica (normalmente incrementa o decrementa) la variabile iniziale.

```
istruzione-for :
  for (inizializzazione; condizione; modifica variabile )
    blocco istruzioni;
```

Volendo schematizzare questo ciclo utilizzando l'istruzione WHILE, ecco la struttura che vi si presenta:

```
inizializzazione variabile;
while ( condizione ){
  istruzione1;
  istruzione2;
  ....
  modifica variabile;
}
```

Come vedete le strutture for e while in molti casi sono intercambiabili. La scelta per l'una o per l'altra dipende molto dello stile del programmatore e dalla leggibilità del programma.

Un esempio: la successione di Fibonacci

Leonardo Fibonacci fu un matematico Pisano vissuto tra 1170 e il 1250. Tra il 1202 e 1220 pubblicò la sua opera di quindici capitoli *Liber Abaci*, tramite la quale introdusse per la prima volta in Europa le nove cifre (da lui chiamate *indiane*), assieme al segno 0, "che in arabo è chiamato zefiro" (cap. I), con confronti con il sistema romano, presentò criteri di divisibilità, regole di calcolo di radicali quadratici e cubici ed altro. Introdusse con poco successo la barretta delle frazioni (nota al mondo arabo prima di lui) (cap. II-IV), ma Fibonacci è soprattutto famoso per la sequenza di numeri interi che da lui prende il nome.

La **successione di Fibonacci** è una sequenza di numeri interi naturali definibile assegnando i valori dei due primi termini, $F_0 := 0$ ed $F_1 := 1$, e chiedendo che per ogni successivo sia $F_n := F_{n-1} + F_{n-2}$. Il termine F_0 viene aggiunto nel caso si voglia fare iniziare la successione con 0; storicamente il primo termine della successione è $F_1 := 1$.

In altri termini dati due numeri interi iniziali (0 e 1 o 1 e 1) i numeri successivi della sequenza sono dati dalla somma di due numeri immediatamente precedenti.

La successione di Fibonacci possiede moltissime proprietà di grande interesse. Certamente la proprietà principale, e maggiormente utile nelle varie scienze, è quella per cui il rapporto F_n / F_{n-1} al tendere di n all'infinito tende al numero algebrico irrazionale chiamato sezione aurea o numero di Fidia

$$\phi = \frac{1 + \sqrt{5}}{2} = 1,6180339887$$

Curioso anche notare come, salendo via via per la sequenza, questo rapporto risulti alternatamente maggiore e minore della costante limite. Naturalmente il rapporto tra un numero di Fibonacci e il suo successivo tende al reciproco della sezione aurea.

$$1/\phi = 0,6180339887...$$

Conviene anche ricordare che:

$$a) \quad \phi - 1 = 1/\phi$$

$$b) \quad 1 - \phi = -1/\phi = \frac{1 - \sqrt{5}}{2}$$

in accordo con la definizione di sezione aurea come il numero positivo tale che

$$\phi = 1 + 1/\phi$$

equazione che, quando vincolata alla condizione $\phi > 0$, ammette l'unica soluzione:

$$\phi = \frac{1 + \sqrt{5}}{2}$$

Si noti poi come l'inverso del reciproco del numero di Fidia $-1/\phi$ che compare nella b) costituisca la seconda soluzione, a segno negativo, dell'equazione algebrica riportata nella definizione. Esso espone proprietà altrettanto interessanti di quelle del suo omologo positivo.

Ragionamenti analoghi possono essere applicati per ottenere altri rapporti irrazionali costanti; per esempio dividendo ogni numero per il secondo successivo si ottiene 0.382 e dividendo ogni numero per il terzo successivo si ottiene 0.236, mentre dividendo ogni numero per il secondo precedente si ottiene 2.618 e dividendo ogni numero per il terzo precedente si ottiene 4.236.

Questi sei rapporti (0,236, 0,382, 0,618, 1, 1,618, 2,618, 4,236) sono inoltre molto utilizzati nella teoria delle onde di Elliott, argomento molto importante dell'analisi tecnica dei mercati finanziari

Altra proprietà interessante è che se un qualsiasi numero della successione è elevato al quadrato, questo è uguale al prodotto tra il numero che lo precede e quello che lo segue, aumentato o diminuito di una unità: per esempio $52 = 25 = 3 * 8 + 1$ oppure $32 = 9 = 2 * 5 - 1$.

Si trova anche che l' n -simo numero di Fibonacci si può esprimere con la formula:

$$F(n) = \frac{\phi^n}{\sqrt{5}} - \frac{(1-\phi)^n}{\sqrt{5}} = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}$$

Tale formula, di rara bellezza ed eleganza, porta il nome di Jacques Binet, che la ricavò nel 1843; tuttavia essa era già nota nel XVIII secolo ad Eulero, Abraham De Moivre e Daniel Bernoulli.

Ecco un semplice programma per scrivere sullo schermo N numeri della sequenza di Fibonacci.

Prima scriveremo il programma in uno pseudolinguaggio, cercando cioè di astrarci da uno specifico linguaggio di programmazione. In seguito vedremo di adattare il nostro programma ad Action Script e alle caratteristiche di FLASH.




```

//scrive la serie di Fibonacci
intero n1 = 0; //contiene il numero n-2 della serie
intero n2 = 1; // contiene il numero n-1 della serie
intero n3; //numero di Fibonacci che viene calcolato
stringa elenco = ""; //Stringa di testo che conterrà l'elenco
                        //dei numeri di Fibonacci
scrivi sullo schermo "Inserisci un numero maggiore di 2";
intero A = - numero inserito dall'utente -

if (A maggiore o uguale 2) {
    //se il numero inserito è maggiore di due elenco conterrà
    //all'inizio i primi due numeri della serie
    elenco = "0, 1, ";
}

for (intero i = 0; i minore di (A-2); incrementa i) {
    //il numero di viene calcolato sommando i due numeri
    //precedenti nella serie
    n3 = n2 + n1;
    //in n1 e n2 vengono ora memorizzati gli ultimi due numeri
    //della serie
    n1 = n2;
    n2 = n3;
    //a elenco viene aggiunto il numero di Fibonacci così
    //ottenuto più una virgola e uno spazio
    elenco = elenco + n3 + ", ";
}
// il ciclo continua fino a che non sono stati creati tutti
// numeri richiesti dell'utente.

// la stringa così ottenute viene scritta sullo schermo
scrivi sullo schermo elenco;


```

La successione di Fibonacci in Action Script

Per realizzare questo programma in Flash con Action Script come prima cosa dovremo creare sullo stage tre campi di testo. Per fare questo utilizzeremo l'apposito tool presente nella *palette strumenti* e utilizzando la *palette proprietà* definiremo le caratteristiche dei campi.

Il primo campo sarà di tipo *testo statico* e qui inseriremo il testo *"Inserisci un numero maggiore di 2"*. Il secondo sarà di tipo *testo di input* e servirà a raccogliere il numero inserito dall'utente: come font del campo sceglieremo *_sans* e gli daremo il nome *input_txt*. Il terzo sarà un campo di tipo *testo dinamico* e servirà a mostrare sullo schermo l'output del programma ossia la serie di Fibonacci che avremo generato: come font del campo sceglieremo *_sans* e gli daremo il nome *messaggio_txt*.

A questo punto premendo F5 aggiungeremo un fotogramma al nostro filmato in modo che sia composto da almeno due fotogrammi e aggiungeremo un livello che chiameremo *Azioni*.

Selezioniamo il primo frame sul livello azioni e apriamo la finestra delle azione cliccando sulla palette delle proprietà il simbolo . Sulla barra della finestra delle azione dovrà essere scritto: *Azioni – Fotogramma* ad indicare che stiamo modificando le azioni per quel fotogramma.

Inseriamo il codice seguente:

```
//scrive la serie di Fibonacci
var n1:Number = 0; //elemento n-2 della serie
var n2:Number = 1; //elemento n-2 della serie
var n3:Number = 0; //numero di Fibonacci calcolare

var elenco:String= ""; //Stringa di testo che conterrà
                        //l'elenco dei numeri di Fibonacci

var A:Number = input_txt.text; //numero inserito dall'utente

if (A >= 2) {
    //se il numero inserito è maggiore di u uguale a 2 elenco
    //conterrà all'inizio i primi due elementi della serie
    elenco = "0, 1, ";
}
for (var i:Number = 0; i < (A-2); i++) {
    //il numero di viene calcolato sommando i due numeri
    //precedenti nella serie
    n3 = n2 + n1;
    //in n1 e n2 vengono ora memorizzati gli ultimi due numeri
    //della serie
    n1 = n2;
    n2 = n3;
    //a elenco viene aggiunto il numero di Fibonacci così
    //ottenuto Più e una virgola e uno spazio
    elenco = elenco + n3 + ", ";
}
// il ciclo continua fino a che non sono stati creati tutti
// i numeri richiesti dell'utente.

// la stringo così ottenute viene scritta nel campo di testo
messaggio_txt.text = elenco;
```

Scegliamo sul menu *controllo* e quindi *prova filmato*. Quando si inserisce un numero nel campo di input viene visualizzate la sequenza di Fibonacci relativa. È possibile trovare il filmato che contiene questo codice nella cartella Esempi del CD il nome del file è *Iterazione_for fla*.

Lo stesso programma è stato implementato anche usando while al posto di for il file relativo è *Iterazione_while fla*.

NB: Il filmato deve essere composto di almeno due fotogrammi perché solo in questo caso gira indefinitamente in *loop*. Ogni volta che si torna al primo fotogramma, il codice che controlla quanto inserito dall'utente viene ripetuto e il contenuto dei campi aggiornato di conseguenza.

Uso dei cicli for..in

ActionScript offre un'ulteriore struttura iterativa il ciclo for..in. Un ciclo for..in consente di eseguire *iterazioni* scorrendo gli elementi secondari di un clip filmato, le proprietà di un oggetto o gli elementi di un array. Gli elementi secondari di un clip filmato sono costituiti da tutto ciò che un clip filmato può contenere: altri clip filmato, funzioni, oggetti e variabili. L'uso più comune di un ciclo for..in è scorrere le proprietà (le coppie valore/chiave) di un oggetto.

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]);  
}
```

Funzioni e metodi

Quando sviluppiamo un programma capita spesso che alcune operazioni debbano essere ripetute più di una volta. Siccome scrivere tutte le volte le medesime operazioni risulterebbe tedioso quanto inutile ci viene in aiuto il concetto di funzione che altro non è che un programmino (o modulo) a se stante il quale prende in entrata dei valori e restituisce un risultato. Altro vantaggio è quello di poter modificare la struttura della funzione (per ottimizzarla o renderla più veloce) senza per questo dover intaccare il codice che la richiama.

Una funzione ha bisogno di essere **dichiarata** e **definita**; cioè vanno specificati i tipi di ingresso e di uscita sui quali la funzione andrà a compiere le proprie operazioni (DICHIARAZIONE) e successivamente dovremo scrivere il "corpo" della funzione vera e propria (DEFINIZIONE).

D'ora in poi ci riferiremo in specifico da ActionScript anche se quando vedremo vale anche per altri linguaggi: innanzi tutto per JavaScript, che ha una sintassi identica, ma, con minime modifiche, anche per JAVA, C++ e C#.

Funzioni e metodi in ActionScript

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`.

Funzioni incorporate

Nel linguaggio `ActionScript` sono incorporato numerosi funzioni che consentono di eseguire determinate attività e di accedere alle informazioni.

Si può trattare di funzioni globali o di funzioni appartenenti ad una classe incorporata nel linguaggio, quale ad esempio `Math` o `MovieClip`, e che, quindi, verranno utilizzate come metodi della classe o dell'oggetto istanza di quella classe. Chiariremo meglio questi concetti quando parleremo delle Classi.

Si può, ad esempio, ottenere il tempo passato da quando un file `SWF` è stato lanciato 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.

Le funzioni di primo livello sono di facile utilizzo. Per chiamare una funzione, è sufficiente utilizzarne il nome e passare tutti i parametri richiesti. Se, ad esempio, aggiungo il codice `ActionScript` seguente al fotogramma 1 della linea temporale:

```
trace("Hello world!");
```

Quando si prova il file `SWF`, verrà visualizzato `Hello worls!` nel pannello `Output`. La funzione `trace`, infatti non fa altro che scrivere un messaggio sulla finestra di `output` e non ritorna alcun valore.

Funzioni utente

Le funzioni utente sono funzioni create dall'utente e che normalmente consentono di compiere operazioni non previsto dal linguaggio e dalle funzioni incorporate. Esistono due modi per dichiarare e definire le funzioni utente: posso scrivere funzioni con nome o funzioni anonime.

Scrittura di funzioni con nome

La funzione con nome è quella più comunemente usata. Quando si compila 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.

La sintassi con cui definisco una funzione con nome è la seguente:

```
function nomefunzione (parametro1, parametro2, ....) {
    // Blocco di istruzioni
}
```

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.
- `parametro1`, `parametro2`, ... uno o più parametri che vengono passati alla funzione. I parametri sono detti anche *argomenti*.
- Blocco di istruzioni 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 istruzioni` è un segnaposto che indica dove deve essere inserito il blocco della funzione.

Proviamo ora a scrivere una funzione utente. La funzione che scriveremo e chiameremo `my_timer` avrà la semplice funzione di scrivere in un campo di testo il tempo passato da quando il filmato è stato aperto. Per farlo utilizzerà la funzione incorporata `getTimer()`.

Come prima cosa dovremo creare sullo stage due campi di testo. Per fare questo utilizzeremo l'apposito tool presente nella *palette strumenti* e utilizzando la *palette proprietà* definiremo le caratteristiche dei campi.

Il primo campo sarà di tipo *testo statico* e qui inseriremo il testo *"Dal via sono trascorsi "*. Il secondo sarà un campo di tipo *testo dinamico* e servirà a mostrare sullo schermo l'output del programma ossia i millisecondi trascorsi: come font del campo sceglieremo `_sans` e gli daremo il nome `messaggio_txt`.

A questo punto aggiungeremo un livello nella linea temporale che chiameremo *Azioni*.

Selezioniamo il primo frame sul livello azioni e apriamo la finestra delle azione cliccando sulla palette delle proprietà il simbolo . Sulla barra della finestra delle azione dovrà essere scritto: *Azioni - Fotogramma* ad indicare che stiamo modificando le azioni per quel fotogramma della linea temporale.

Inseriamo il codice seguente:

```
stop();
//semplice timer che misura il tempo trascorso
//dal lancio del filmato

//definisco la funzione my_timer() che non ritorna alcun
//valore e che aggiorna il contenuto del campo di testo
```

```
//messaggio_txt sulla base di quanto restituito dalla
//funzione incorporata getTimer

function my_timer ():Void {
    //millisecondi trascorsi
    var t:Number = getTimer();
    //secondi = parte intera delle divisione
    var s:Number = Math.floor(t/1000);
    //millesimi = resto della divisione
    var m:Number = t % 1000;

    messaggio_txt.text = s + " secondi e " + m +
        " millesimi.";
}
// la funzione my_timer viene eseguita una prima volta
// appena il filmato viene eseguito

my_timer();

// poi utilizzando il timer del computer la funzione
// my_timer viene eseguita ogni 30 millisecondi

setInterval(my_timer, 30);
```

L'esempio qui riportato si trova nel file FunzioneConNome fla.

Per creare la propria funzione in ActionScript si utilizza l'istruzione `function`. 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 sono normalmente scritte sulla linea temporale del filmato principale o all'interno di file ActionScript esterni, ad esempio file di classe.

Un particolare tipo di funzione è la funzione di costruzione di una classe detta *constructor*. In ActionScript queste funzioni usano la stessa sintassi delle normali funzioni utente. Unica prescrizione è che il nome della funzione deve corrispondere al nome della classe. Vedremo meglio come scrivere funzioni, (cioè metodi) nelle classi, quando parleremo di classi.

Scrittura di funzioni anonime e di callback

Esiste un modo alternativo per definire una funzione: quello di creare una funzione anonima.

Il costrutto tipico con cui definisco una funzione anonima è il seguente:

```
var nomevariabile = function (parametro1, parametro2, ....) {
    // Blocco di istruzioni
}
```

Questa porzione di codice comprende le parti seguenti:

- nomevariabile è il nome di una variabile.
- parametro1, parametro2, ... uno o più parametri che vengono passati alla funzione. I parametri sono detti anche *argomenti*.
- Blocco di istruzioni contiene tutto il codice ActionScript relativo alla funzione. Questa parte contiene le istruzioni che eseguono le azioni, ovvero il codice che si desidera eseguire.

Dopo che la funzione è stata definita e il suo contenuto assegnato ad una variabile posso usare la variabile esattamente come se avessi definito una funzione con nome.

Prendiamo l'esempio precedente. Possiamo ottenere il medesimo risultato utilizzando una funzione anonima:

```
stop();
//semplice timer che misura il tempo trascorso
//dal lancio del filmato - utilizza una funzione anonima

// assegno alla variabile my_timer una funzione
// e che aggiorna il contenuto del campo di testo
// messaggio_txt sulla base di quanto restituito dalla
// funzione incorporata getTimer

var my_timer:Function = function ():Void {
    //millisecondi trascorsi
    var t:Number = getTimer();

    //secondi = parte intera delle divisione per 1000
    var s:Number = Math.floor(t/1000);

    //millesimi = resto della divisione per 1000
    var m:Number = t % 1000;

    messaggio_txt.text = s + " secondi e " + m +
        " millesimi.";
}
// la funzione my_timer viene eseguita una prima volta
// appena il filmato viene eseguito

my_timer();

// utilizzazndo il timer del computer la funzione my_timer
// viene
// eseguita ogni 30 millisecondi
setInterval(my_timer, 30);
```

L'esempio qui riportato si trova nel file FunzioneAnonima_1 fla.

LA PROGRAMMAZIONE ITERATIVA

Notate la differenza tra `function` (con la iniziale minuscola) che è l'istruzione che consente di definire una funzione e `Function` (con la iniziale maiuscola) che indica il tipo `Function` e comunica al compilatore che la variabile `my_timer` conterrà una funzione. In `ActionScript` non è obbligatorio, quando si dichiara una variabile (con l'istruzione `var`), dichiarare il tipo di dati che la variabile è destinata a contenere, ma è fortemente consigliato.

Nel caso illustrato posso usare indifferentemente una funzione anonima o una funzione con nome: il risultato pratico è praticamente identico. Normalmente viene scelta la prima versione (funzione con nome) perché più leggibile.

In realtà c'è un po' di differenza tra l'uso di una funzione con nome e di una variabile che contiene una funzione anonima. Se per esempio modifico il codice in cui uso la funzione con nome mettendo la definizione della funzione infondo così:

```
stop();
//semplice timer che misura il tempo trascorso dal lancio del filmato

// la funzione my_timer viene eseguita una prima volta
// appena il filmato viene eseguito

my_timer(); // funziona!!!!

// poi utilizzando il timer del computer la funzione
// my_timer viene eseguita ogni 30 millisecondi

setInterval(my_timer, 30);

//definisco la funzione my_timer() che non ritorna alcun valore e che aggiorna il
//contenuto del campo di testo messaggio_txt sulla base di quanto restituito dalla
//funzione incorporata getTimer

function my_timer ():Void {
    //millisecondi trascorsi
    var t:Number = getTimer();
    //secondi = parte intera delle divisione
    var s:Number = Math.floor(t/1000);
    //millesimi = resto della divisione
    var m:Number = t % 1000;
    messaggio_txt.text = s + " secondi e " + m + " millesimi.";
}
```

Il mio codice funzionerà lo stesso. Mentre se apporto una modifica simile al codice in cui ho usato la funzione anonima, così:

```
stop();
//semplice timer che misura il tempo trascorso dal lancio del filmato

//la funzione my_timer viene eseguita una prima volta appena il filmato viene eseguito

my_timer(); //errore!!!!

// utilizzando il timer del computer la funzione my_timer viene
// eseguita ogni 30 millisecondi
setInterval(my_timer, 30);

//assegno alla variabile my_timer una funzione che aggiorna il contenuto del campo di
//testo messaggio_txt sulla base di quanto restituito dalla funzione getTimer

var my_timer:Function = function ():Void {
    //millisecondi trascorsi
    var t:Number = getTimer();

    //secondi = parte intera delle divisione per 1000
    var s:Number = Math.floor(t/1000);

    //millesimi = resto della divisione per 1000
    var m:Number = t % 1000;
    messaggio_txt.text = s + " secondi e " + m +
        " millesimi.";
}
```

Il compilatore mi riporterà un errore. Nel primo caso `my_timer` è una funzione e viene compilata (come se fosse un modulo separato) prima dell'altro codice. Nel secondo caso `my_timer` è una variabile e non contiene nulla fino a che non gli è stato assegnato un valore.

Ci sono casi in cui l'uso delle funzioni anonime è necessario. Il caso più tipico è quando devo gestire un evento legato ad un oggetto. In questo la sintassi è la seguente:

```
oggetto.evento = function (parametro1, parametro2, ...) {
    // Blocco di istruzioni che gestiscono l'evento
}
```

Questa porzione di codice comprende le parti seguenti:


- `oggetto` è il nome di oggetto il cui evento deve essere gestito.
- Evento è un evento che l'oggetto *spara* in determinate condizioni (nel caso di un bottone può essere, ad esempio, `onRelease`, `onPress`, `onRollOver`, ecc.)
- `parametro1`, `parametro2`, ... uno o più parametri che vengono passati alla funzione gestore d'evento. I parametri sono detti anche *argomenti*.
- Blocco di istruzioni contiene tutto il codice ActionScript che deve essere eseguito quando si verifica l'evento.

Nell'esempio seguente scriveremo il codice che carica un suono, lo esegue quando il caricamento è terminato e segnala con appositi messaggi sullo schermo quando il suono inizia e finisce.

Come prima cosa dovremo creare sullo stage un campo di testo. Per fare questo utilizzeremo l'apposito tool presente nella *palette strumenti* e utilizzando la *palette proprietà* definiremo le caratteristiche del campo.

Il campo sarà un campo di tipo *testo dinamico* e servirà a mostrare sullo schermo l'output del programma ossia i messaggi che segnaleranno che il suono è iniziato e terminato: come font del campo sceglieremo `_sans` e gli daremo il nome `messaggio_txt`.

A questo punto aggiungeremo un livello nella linea temporale che chiameremo *Azioni*.

Selezioniamo il primo frame sul livello azioni e apriamo la finestra delle azioni cliccando sulla palette delle proprietà il simbolo . Sulla barra della finestra delle azioni dovrà essere scritto: *Azioni - Fotogramma* ad indicare che stiamo modificando le azioni per quel fotogramma della linea temporale.

Inseriamo il codice seguente:

```

stop();
//uso di una funzione anonima per gestire eventi

//creo un'istanza della classe sound

var mio_suono:Sound = new Sound();
//la variabile mio_souno contiene ora un'istanza
//della classe sound

//definisco la funzione che viene eseguita
//quando il suono viene caricato completamente
mio_suono.onLoad = function(success) {
    if (success) {
        //aggiorno il testo
        messaggio_txt.text = "Il suono è in esecuzione."
        //faccio partire il suono
        this.start();
    } else {
        //se success è falso significa che c'è
        //stato un errore
        messaggio_txt.text = "Si è verificato un errore!"
    }
}

//definisco la funzione che viene eseguita
//quando l'esecuzione del suono è completata
mio_suono.onSoundComplete = function() {
    //aggiorno il testo
    messaggio_txt.text = "Esecuzione completata.";
}

//carico il suono nell'oggetto mio_souno
mio_suono.loadSound("Round.mp3", false);

```

L'esempio qui riportato si trova nel file FunzioneAnonima_2 fla.

Definizione di funzioni globali e associate alla linea temporale
 Le funzioni sono associate alla linea temporale del clip filmato che le definisce. È quindi necessario utilizzare un percorso target per chiamarle.

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 definire una funzione anonima e assegnarla a una variabile preceduta dal percorso `_global` come nell'esempio seguente:

```

_global.myFunction = function(myNum:Number):Number {
    return (myNum * 2) + 3;
};
trace(myFunction(5)) // 13

```

Dopo che è stata definita la funzione `myFunction` potrà essere richiamata da qualsiasi riga di codice `ActionScript` senza specificare alcun percorso target come succede per le funzioni incorporate.

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));
```

La funzione sarà disponibile a partire dal frame in cui è stata definita e potrà essere richiamata senza specificare un percorso target dal `MovieClip` corrente. Per richiamare la funzione da un altro `MovieClip` sarà necessario specificare un percorso target relativo o assoluto.

Se, ad esempio, la funzione `areaOfCircle` è stata definita nel `MovieClip` `mycircle_mc` che è collocato nel filmato principale, potrà essere raggiunta (percorso target assoluto) scrivendo:

```
var r:Number = 7;
var Area:Number = root.mycircle_mc.areaOfCircle(r);
```

Uso di variabili nelle funzioni

Le variabili locali sono strumenti utili per organizzare il codice e renderlo più comprensibile. 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.

Facciamo un esempio:

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.

Come possiamo vedere i parametri, detti anche *argomenti*, sono gli elementi che il codice della funzione elabora per produrre il suo risultato.

Passaggio di parametri a una funzione

Si possono passare più parametri ad una funzione separandoli con delle virgole.

Talvolta i parametri sono obbligatori e talvolta sono facoltativi. In una funzione potrebbero essere presenti sia parametri obbligatori che opzionali.

In ogni caso se si passa alla funzione un numero di parametri inferiore a quelli dichiarati, Flash imposta i valori dei parametri mancanti a undefined. Questo può provocare risultati imprevisti:

1. Creare un nuovo documento Flash e salvarlo come Somma_0 fla.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
function somma(a:Number, b:Number, c:Number):Number {
    return (a + b + c);
}
// sommo tre numeri
trace(somma(1, 4, 6)); // 11
// La somma non è un numero (c è uguale a undefined)
trace(somma(1, 4)); // NaN
// il parametro non dichiarato è ignorato
trace(addNumbers(1, 4, 6, 8)); // 11
```

3. Selezionare Controllo > Prova filmato per provare il documento Flash. Flash visualizza i seguenti valori nella finestra di output: 11, NaN, 11.

Se non si passa un numero sufficiente di parametri alla funzione somma, agli argomenti mancanti verrà assegnato il valore predefinito undefined. Se si passano troppi parametri, quelli in eccesso verranno ignorati.

È possibile scrivere funzioni con numero di parametri variabile. Prendiamo l'esempio in Somma fla:

```
//definisco una funzione somma che somma fra loro
//un numero qualsiasi di numeri interi

function somma():Number {
    var sum:Number = 0;
    // arguments è una speciale variabile che
    // contiene in un array i parametri passati
    // alla funzione
    // somma a sum tutti i numeri passati come parametri
    for (var i = 0; i < arguments.length; i++) {
        // controllo che il parametro passato sia un
        // numero se no lo ignoro
```

```

        if (! isNaN(arguments[i])) {
            sum = sum + arguments[i];
        }
    }
    // restituisco il valore di sum
    return (sum);
}

trace (somma()); // 0
trace (somma(5,6,7,20)); //38
trace(somma(10,7)); //17

trace (somma(10,"pippo",89)) //99;

```

Per scrivere funzioni che possano avere un numero qualsiasi di parametri devo usare nel corpo della funzione al posto dei parametri dichiarati la variabile speciale `arguments` che organizza in un Array i parametri passati alla funzione.

In questi casi i parametri non vengono dichiarati e quindi il controllo del tipo (il controlla, cioè che vengano passati parametri coerenti alle azioni che compie la funzione) sono a carica del programmatore.

Restituzione di valori da funzioni

Una funzione può restituire un valore che di norma è il risultato dell'operazione compiuta. Per compiere questa operazione si utilizza l'istruzione `return` che specifica il valore che verrà restituito dalla funzione.

L'istruzione `return` ha anche l'effetto di interrompere immediatamente il codice in esecuzione nel corpo della funzione e restituire immediatamente il controllo del flusso di programma al codice chiamante.

Nell'utilizzo dell'istruzione `return` si applicano le regole seguenti:

- 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 `return` viene specificata, non deve essere seguita da valori.
- Indipendentemente dal tipo restituito, un'istruzione `return` può essere utilizzata per uscire da una funzione e restituire il controllo al codice chiamante
- 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:

1. Creare un nuovo documento Flash e salvarlo come return fla.
2. Aggiungere il codice seguente al fotogramma 1 della linea temporale principale:

```
//La funzione getArea accetta due parametri: width e height.
//entrambi numeri

function getArea(width:Number, height:Number):Number {
    return width * 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 nella variabile `area`. Il valore salvato nella variabile `area` vengono quindi scritti sulla finestra di output con il comando `trace`.

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.

Nozioni fondamentali sui metodi

I metodi sono funzioni associate a una classe che può essere una classe scritta da voi o incorporata, ovvero parte del linguaggio `ActionScript`. Le funzioni incorporate in una classe si chiamano metodi perché sono strumenti che ci fornisce l'oggetto per svolgere il suo compito.

Vediamo ora un esempio di come si utilizza un metodo di una classe incorporata. Per farlo useremo alcuni metodi associati alla classe `Array`:

1. Creo un nuovo documento Flash e salvarlo come metodi fla.
2. Aggiunge il codice seguente nel primo fotogramma 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 creato un nuovo oggetto Array denominato userArr. Viene utilizzato il metodo push() della classe Array per aggiungere all'array tre oggetti che, a loro volta, contengono nome ed età. Utilizzando il metodo sortOn() l'array viene ordinato in base al valore della proprietà firstname di ogni oggetto. Utilizzando un ciclo di iterazione gestito con for, si visualizza il nome di ogni elemento dell'array nel pannello Output. I nomi vengono visualizzati in ordine alfabetico.

3. Selezionare Controllo > Prova filmato per provare il file SWF.

Le classi

I linguaggi orientati agli oggetti si basano sul concetto di *classi* e *interfacce*. Una classe definisce tutte le proprietà e i comportamenti che distinguono una serie di oggetti.

Se vogliamo organizzare i diversi utenti che usano un'applicazione possiamo creare una classe *User*, ad esempio, che rappresenta gli utenti. Nella definizione della classe vengono definite le proprietà che hanno gli utenti (ad esempio nome, cognome, username, password, ecc.). Per gestire i singoli utenti vengono create istanze della classe, che, per la classe *User*, rappresentano ogni singolo individuo, ovvero, come si dice, i membri della classe.

Le istanze della classe *User* comprendono tutte le proprietà della classe *User*.

Quando abbiamo parlato dei tipi di dati abbiamo visto che esistono tipi di dati primitivi e tipi di dati derivati o complessi. In linea di principio in un linguaggio orientato agli oggetti tutti i tipi di dati sono classi. Creando una nuova classe il programmatore crea un nuovo tipo di dati complesso, un modello, cioè, delle proprietà e delle caratteristiche che ha un nuovo tipo di oggetto. Per creare un tipo di dati *Lattuga* dovrò scrivere la classe *Lattuga* all'interno della quale definirò ad esempio le proprietà *tipo_foglia* e *colore* e il metodo *lavare()*. In questo modo si definisce l'oggetto *Lattuga* alle cui istanze potranno essere applicato il metodo *lavare()* e potranno essere impostate le proprietà *tipo_foglia* e *colore*.

Nozioni di base

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).

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.

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.

La classe Gatto di cui abbiamo parlato potrebbe essere l'estensione di una superclasse dalla quale eredita da proprietà e metodi. 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.

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 programmatori, 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

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.

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.

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.

Le classi in ActionScript

Prima di passare alla programmazione di classi personalizzate vedremo di familiarizzare con il concetto di classe imparando ad utilizzare le classi incorporate in ActionScript.

Nella scrittura del codice ActionScript segue questa convenzioni: i nomi delle variabili, delle proprietà, delle funzioni e dei metodi iniziano sempre con una lettera minuscola o, a volte con "_" (underscore), i nomi dei tipi di dati e, quindi, delle classi con una lettera maiuscola.

ActionScript comprende circa 65 classi incorporate che forniscono diversi tipi di elementi: tipi di dati di base quali Array, Boolean, Date e così via, gestione degli errori, gestione degli eventi, caricamento di contenuto esterno (XML, immagini, dati binari originari e così via).

Nella maggior parte dei casi utilizzare una classe significa creare un'istanza della classe stessa, assegnare l'istanza creata ad una variabile e usarne le proprietà e applicarne i metodi per ottenere i risultati desiderati. Per creare un'istanza di una classe devo applicare l'operatore new al constructor, cioè alla funzione di costruzione della classe, una speciale funzione che ha lo stesso nome della classe.

Analizziamo l'esempio che segue:

```
// dichiaro "now" come variabile di tipo Date
var now:Date;

// assegno a now una istanza di Date che corrisponde alla
// data e all'ora corrente

now = new Date();
```

Dopo questa operazione `now` conterrà un'istanza della classe `Date` che rappresenta la data e l'ora corrente. A `now` si potranno applicare tutti i metodi e usare le proprietà della classe `Date`.

Da notare l'uso diverso di `Date` in `var now:Date;` e in `now = new Date();`. Nel primo caso `Date` indica il tipo `Date` e il costrutto dice al compilatore che la variabile `now` potrà contenere solo oggetti di tipo `Date`. Nel secondo caso `Date()` (seguito da parentesi) indica la funzione di costruzione della classe `Date` e il costrutto `now = new Date()` crea una nuova istanza della classe `Date()` e la memorizza in `now`.

Questa procedura è comune a buona parte delle classi. Ci sono però importanti eccezioni:

- L'operatore `new` non deve essere usato per i tipi primitivi: `Number`, `String`, `Boolean` (che comunque sono classi a tutti gli effetti, con le loro proprietà e i loro metodi). Le istanze delle classi sono create automaticamente quando maneggio questo tipo di dati.
- Ci sono delle classi che non hanno membri (per cui, quindi, non si possono creare istanze) e che rappresentano in quanto tali un oggetto su cui operare: `Accessibility`, `Camera`, `ContextMenu`, `CustmActions`, `Key`, `Math`, `Microphone`, `Mouse`, `Selection`, `SharedObject`, `Stage`, `System`. In questi casi proprietà e metodi si applicano direttamente alla classe e vengono detti statici.
- Infine ci sono classi la cui istanze vengono create automaticamente quando creo il o colloco il simbolo corrispondente in un fotogramma del filmato. Queste classi sono `Button`, `MovieClip`, `TextField` e `Video`.

Qui di seguito trovate l'elenco delle classi incorporate in `ActionScript`.

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 _accProps per controllare le proprietà accessibili di clip filmato, pulsanti e campi di testo in fase di runtime.
Array	La classe Array rappresenta gli array di <code>ActionScript</code> ; tutti gli oggetti array sono istanze di questa classe. La classe <code>Array</code> fornisce i metodi e le proprietà per eseguire operazioni con oggetti <code>Array</code> .
AsBroadcaster	Fornisce funzionalità di notifica eventi e gestione listener che possono essere aggiunte ad altri oggetti.
Boolean	La classe <code>Boolean</code> è un wrapper per i valori booleani (<code>true</code> o <code>false</code>).
Button	La classe <code>Button</code> fornisce metodi, proprietà e gestori di eventi per eseguire le operazioni con i pulsanti.
Camera	La classe <code>Camera</code> fornisce accesso alla videocamera dell'utente, se installata. Se utilizzato con <code>Flash Communication Server</code> , il file SWF può acquisire, trasmettere e registrare immagini e video provenienti da una videocamera.

Classe	Descrizione
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 (<code>flash.geom.ColorTransform</code>).
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 .
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.
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.
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 throw che è possibile poi gestire tramite l'istruzione try..catch..finally .
Function	La classe Function è la rappresentazione di classe di tutte le funzioni ActionScript, comprese quelle native di ActionScript e quelle definite dall'utente.
Key	La classe Key fornisce i metodi e le proprietà per accedere alle informazioni sulla tastiera e i tasti premuti.
LoadVars	La classe LoadVars consente di trasferire variabili tra un file SWF e un server in coppie nome/valore.
LocalConnection	La classe LocalConnection consente di sviluppare file SWF in grado di inviare istruzioni gli uni agli altri senza utilizzare il metodo fscommand() di JavaScript.
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 Math.metodo(parametro) o Math.costante .
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.
Mouse	La classe Mouse consente di controllare il mouse in un file SWF; tale classe consente, ad esempio, di mostrare o nascondere il puntatore.
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 il clip filmato.
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.
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.
NetStream	La classe NetStream controlla la riproduzione dei file FLV da un file system locale o un indirizzo HTTP.
Number	La classe Number è un wrapper per il tipo di dati numerico di base.

Classe	Descrizione
Object	La classe Object si trova alla radice della gerarchia di classi ActionScript; tutte le altre classi ereditano i suoi metodi e proprietà.
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.
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.
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.
Sound	La classe Sound consente di controllare l'audio di un file SWF.
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.
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.
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.
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.
TextFormat	La classe TextFormat consente di applicare gli stili di formattazione ai caratteri o paragrafi di un oggetto TextField.
TextSnapshot	L'oggetto TextSnapshot consente di accedere a testo statico ed eseguirne il layout all'interno di un clip filmato.
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).
XML	Questa classe fornisce i metodi e le proprietà per eseguire operazioni con oggetti XML.
XMLNode	La classe XMLNode rappresenta un singolo nodo in un albero di documento XML. È la superclasse della classe XML.
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.
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).

Nei capitoli seguenti vedremo come si usano alcune di queste classi incorporate.

Object, Array e Date

Iniziamo il nostro percorso tra le classi incorporate nel linguaggio ActionScript da tre classi di base presenti, con piccole differenze in tutti i linguaggi orientati agli oggetti. Per informazione sulle altre classi incorporate rimandiamo a “Guida di riferimento di ActionScript 2.0”, presente sul CD allegato alla dispensa.

Object

Come abbiamo visto un oggetto è un insieme di proprietà. Una *proprietà* è un attributo che descrive lo stato dell'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();
```

Normalmente è la classe che definisce metodi e proprietà degli oggetti membri.

Il tipo di dati `Object` rappresenta l'astrazione dell'idea di oggetto. Sta alla radice della gerarchia delle classi: tutte le classi sono estensioni di `Object`.

Oltre a definire le proprietà e i metodi base di ogni classe, la classe `Object` mi consente di creare oggetti vuoti, senza proprietà né metodi. Creando

un'istanza di Object creo un oggetto personalizzato, a cui posso aggiungere le proprietà che voglio e che mi può servire per organizzare le informazioni nell'applicazione Flash.

In una applicazione di shopping on-line potrei avere bisogno di 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";
```

Lo stesso oggetto può essere creato anche assegnando alla variabile il letterale di tipo Object corrispondente.

```
var user:Object;
user = {name:"Irving",age:32,phone:"555-1234"};
```

In questa sintassi non è necessario richiamare il costruttore della classe Object() con l'operatore new. In questo corso, comunque, privilegeremo sempre la prima sintassi perché più chiara e coerente con l'utilizzo delle altre classi.

Un altro esempio interessante dell'uso del tipo Object lo trovi in metodi.fla:

```
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);
}
```

Qui viene usata la sintassi abbreviata di creazione di un oggetto per assegnare gli oggetti ad elementi di un Array.

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, come per il tipo Object, è possibile assegnare direttamente alla variabile un letterale di tipo Array:

```
var myArr:Array = ["January", "February", "March", "April"];
```

Un array può essere paragonato a un edificio ad uso uffici, dove ogni piano contiene dati diversi, ad esempio la contabilità al terzo piano e la progettazione 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 posizione di ogni elemento dell'array come ogni piano dell'edificio ha un numero.

Ogni elemento può contenere dati (vale a dire un numero, una stringa, un valore booleano, un array, 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, aggiungere od eliminare un piano. Analogamente si possono spostare i valori in punti diversi dell'array e modificarne le dimensioni.

L'edificio (l'array) contiene piani (gli elementi), ogni piano è numerato (l'indice) e contiene uno o più reparti (i valori).

Potete trovare il file di esempio, `array.fla`, nella cartella Esempi. 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.

Uso degli array

Gli array possono essere utilizzati in modi diversi. Se si caricano dati dal un database collocato su un server Web remoto, probabilmente ci converrà organizzare i dati sotto forma di array di oggetti.

Immaginiamo, ad esempio, di costruire un jukebox: la sequenza dei brani potrebbe essere memorizzata come un array di informazioni sui brani, a loro volta organizzate in oggetti. Ogni oggetto potrebbe contenere il nome del brano, il nome dell'artista, la durata, il percorso del file audio (ad esempio MP3) o altre informazioni che si desidera associare a un determinato brano.

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.

La classe incorporata `Array` consente di accedere agli array e di manipolarli. Per creare un oggetto `Array`, utilizzare la funzione di costruzione `new` con il costruttore `Array()` o l'operatore di accesso agli array `[]`. Nell'esempio seguente viene costruito un array indicizzato.

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.

Nota bene che il medesimo array poteva essere creato utilizzando l'operatore di accesso agli array ([]). In questo caso il codice sarebbe risultato così:

```
// Definisce un nuovo array
var myArr:Array = ["value1","value0"];
// Definisce valori su due indici
// 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]);
}
```

Il costrutto è più sintetico ma meno leggibile e didatticamente meno coerente alla sintassi di creazione delle istanze di una classe, per cui in questo corso utilizzeremo sempre la prima versione.

Modifica di un array

L'array può essere controllato e modificato tramite ActionScript. È possibile spostare valori all'interno dell'array o modificarne la dimensione. Il seguente codice, ad esempio, scambia i dati di due indici di un array:

```
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 perché se il contenuto dell'indice 4 dell'array fosse stato copiato nell'indice 2 dell'array senza salvarne prima il contenuto, il contenuto originale dell'indice 2 sarebbe andato perso.

Con il codice seguente, ad esempio, il valore dell'indice 2 dell'array (Accounting) andrebbe perso:

```
// Modalità scorretta; non viene utilizzata la variabile
temporanea
buildingArr[2] = buildingArr[4];
buildingArr[4] = buildingArr[2];
```

```
trace(buildingArr);
//undefined,undefined,Engineering,undefined,Engineering
```

Lunghezza di un array

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. La proprietà `length` restituisce la lunghezza dell'array.

Qui di seguito troverai due esempi:

Nel primo esempio, viene creato un array che contiene i nomi dei mesi. Vengono visualizzati il contenuto e la lunghezza dell'array. 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.

```
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
```

Nel secondo esempio se si crea un elemento all'indice 5 dell'array, la lunghezza dell'array restituisce 6 (l'array ha base zero) e non il numero effettivo di elementi contenuti nell'array: questo perché gli elementi dall'indice 0 all'indice 4 valgono `undefined`. Se, cioè, assegnando valori ad elementi di un array salto degli indici, gli elementi saltati esisteranno comunque e varranno `undefined`.

```
var myArr:Array = new Array();
myArr[5] = "five";
trace(myArr.length); // 6
trace(myArr);
    //undefined,undefined,undefined,undefined,undefined,5
```

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. Abbiamo appena 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 all'elemento all'indice 5 dell'array viene assegnato lo speciale valore `undefined` anziché essere eliminato totalmente.

Array indicizzati

Come abbiamo visto gli array indicizzati memorizzano una serie di uno o più valori. Gli elementi possono essere recuperati in base alla loro posizione nell'array (indice). Il primo indice è sempre il numero 0 e viene incrementato di un'unità a ogni elemento successivo aggiunto all'array.

Il metodo `push()` della classe array consente di aggiungere elementi ad un array indicizzato.

Vediamo l'esempio seguente:

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. Viene poi chiamato il metodo `push()` per aggiungere elementi all'array.

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

4. Tornare all'ambiente di creazione ed eliminare il codice nel pannello Azioni.
5. Al posto del codice eliminato inserire il codice ActionScript seguente:

```
var myArray:Array = ["one", "two", "three"];
trace(myArray); // one,two,three
```

In questo codice si utilizza il valore letterale array (["one", "two", "three"]) per creare un nuovo array.

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 tabella 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 compiti:Array = ["lavare i piatti", "gettare  
l'immondizia"];
```

Se, ad esempio, 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:

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 (in questo modo il comando non verrà eseguito), 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.

Per creare array multidimensionali è possibile utilizzare cicli for nidificati, come illustrato nell'esempio seguente.

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] = "riga " + i + " colonna " + j;
    }
}
trace(mainArr);
```

Questo codice ActionScript crea un array 3 x 3 e imposta il valore di ogni nodo dell'array. L'array (mainArr) viene quindi tracciato.

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

I dati seguenti saranno visualizzati nel pannello Output: riga 0 colonna 0, riga 0 colonna 1, riga 0 colonna 2, riga 1 colonna 0, riga 1 colonna 1, riga 1 colonna 2, riga 2 colonna 0, riga 2 colonna 1, riga 2 colonna 2

Per eseguire iterazioni su elementi di un array multidimensionale, si procede in maniera simile:

1. Creare un nuovo documento Flash e salvarlo come multiArray3 fla.
2. Copiare dall'esempio precedente il codice seguente e inserirlo al fotogramma 1 della linea temporale:

```
//codice copiato da multiArray2 fla
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] = "riga " + i + " colonna " + j;
    }
}
```

In questo codice, riportato nell'esempio precedente crea un array 3 x 3 e imposta il valore di 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. Il ciclo esterno esegue un'iterazione su ogni elemento di mainArray. Il ciclo interno esegue un'iterazione su ogni array nidificato e fornisce come output ogni nodo dell'array.

4. Selezionare Controllo > Prova. I dati seguenti verranno visualizzati nel pannello Output:

riga 0 colonna 0
 riga 0 colonna 1
 riga 0 colonna 2

riga 1 colonna 0
riga 1 colonna 1
riga 1 colonna 2
riga 2 colonna 0
riga 2 colonna 1
riga 2 colonna 2

Creazione di array associativi

Un array associativo è composto da *chiavi* e *valori* non ordinati e utilizza le chiavi alfanumeriche 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.

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.

Quando si utilizzano array associativi è possibile chiamare l'elemento dell'array desiderato utilizzando una stringa anziché un indice numerico. Dal punto di vista di ActionScript un array associativo è esattamente equivalente a un oggetto generico (tipo Object).

Questo mi consente di vedere, a seconda delle mie necessità un Object come array associativo e viceversa. Vediamo l'esempio seguente:

1. Creare un nuovo documento Flash e salvarlo come arrayAssociativo.fla.
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 = 0xDFDFDF;
// Visualizza una proprietà utilizzando l'operatore punto e
// la sintassi di accesso agli array
trace(someObj.myAlpha); // 72
trace(someObj["myShape"]); // 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 visualizzano su pannello di output due proprietà utilizzando in un caso l'operatore punto e nell'altro la sintassi di accesso agli array.

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

La cosa rilevante messa in evidenza da questo esempio è che posso accedere alle proprietà di qualsiasi oggetto usando indifferentemente la sintassi del punto (`someObj.myColor`) e la sintassi degli array (`someObj['myColor']`) utilizzando quindi il nome della proprietà in formato stringa. Come vedremo questa caratteristica viene spesso usata nel codice che manipola gli oggetti.

Date

La classe `Date` consente di recuperare i valori relativi alla data e all'ora del tempo universale (UTC) o del sistema operativo su cui è in esecuzione Flash Player. Per chiamare i metodi della classe `Date`, è prima necessario creare un oggetto `Date` mediante la funzione di costruzione della classe `Date`.

La funzione di costruzione `Date()` può avere diverse sintassi.

Posso passare alla funzione una data del passato o del futuro. Il questo caso `Date()` richiede da due a sette parametri (anno, mese, giorno, ora, minuti, secondi, millisecondi).

In alternativa, posso costruire un oggetto `Date` passando un singolo parametro che rappresenta il numero di millisecondi trascorsi dal 1 gennaio 1970 alle 0:00:000 GMT.

Se, infine, non specifico alcun parametro all'oggetto data `Date()` vengono assegnate la data e l'ora correnti.

Alcune esempi:

```
var d1:Date = new Date();  
var d3:Date = new Date(2000, 0, 1);  
var d4:Date = new Date(65, 2, 6, 9, 30, 15, 0);  
var d5:Date = new Date(-14159025000);
```

Nella prima riga di codice, un oggetto `Date` viene impostato sull'ora e la data correnti.

Nella seconda riga, viene creato un oggetto `Date` a cui vengono passati i parametri `year`, `month` e `date`, con la data e l'ora corrispondenti alle 0:00:00 GMT del 1 gennaio 2000.

Nella terza riga, viene creato un oggetto Date a cui vengono passati i parametri year, month e date, con la data e l'ora corrispondenti alle 09:30:15 GMT (+ 0 millisecondi) del 6 marzo 1965. Poiché il parametro year è specificato sotto forma di intero a due cifre, viene interpretato come 1965.

Nella quarta riga, viene passato solo un parametro, ovvero un valore temporale che rappresenta il numero di millisecondi prima o dopo le 0:00:00 GMT del 1 gennaio 1970; dal momento che il valore è negativo, rappresenta un tempo *precedente* alle 0:00:00 GMT del 1 gennaio 1970 e in questo caso l'ora e la data sono le seguenti: 02:56:15 GMT del 21 luglio 1969.

Vediamo ora come si utilizza l'oggetto Date. Nell'empio che segue realizzeremo un semplice orologio-datario.

4. Creare un nuovo documento Flash e salvarlo come Date_1 fla.
5. Creare sullo stage un campo di testo dinamico. Scegliere come font del campo _sans e dargli il nome messaggio_txt.
6. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
stop();
//semplice orologio-calendario perpetuo
//utilizza setInterval()
//creo un array mesi che contiene i nomi dei mesi
var mesi:Array = new Array("gennaio", "febbraio", "marzo",
    "aprile", "maggio", "giugno", "luglio", "agosto",
    "settembre", "ottobre", "novembre", "dicembre");

//e un array giorni che contiene i nomi dei giorni della
    settimana
var giorni:Array = new Array("domenica", "lunedì", "martedì",
    "mercoledì", "giovedì", "venerdì", "sabato");

//scrivo una semplice funzione che aggiunge uno "0"
//davanti ad un numero se è composto da una sola cifra
//e lo trasforma in stringa
function zeroPrima(n:Number):String {
    var s:String = n.toString();
    if (s.length == 1) {
        s = "0" + s;
    }
    return (s);
}

//definisco la funzione aggiornaOrologio() che non ritorna
//alcun valore e che aggiorna il contenuto del campo di
//testo messaggio_txt sulla base di quanto restituito
//dall'oggetto now
```

```
function aggiornaOrologio ():Void {
    //creo un oggetto date che contiene data e ora corrente
    var now:Date = new Date();

    var giorno_sett:String = giorni[now.getDay()];
    var giorno:Number = now.getDate();
    var mese:String = mesi[now.getMonth()];
    var anno:Number = now.getFullYear();
    var ora:String = zeroPrima(now.getHours());
    var minuti:String = zeroPrima(now.getMinutes());
    var secondi:String = zeroPrima(now.getSeconds());
    messaggio_txt.text = giorno_sett + " " + giorno + " " +
    mese + " " + anno + " " + ora + ":" + minuti + ":" +
    secondi;
}
// la funzione aggiornaOrologio viene eseguita una prima
// volta appena il filmato viene eseguito

aggiornaOrologio();

// poi utilizzando il timer del computer la funzione
// aggiornaOrologio viene eseguita ogni 100 millisecondi

setInterval(aggiornaOrologio, 100);
```

Il codice crea due array che contengono rispettivamente i nomi dei mesi e dei giorni della settimana in lingua italiana. Poi definisce una funzione `zeroPrima()` che trasforma un numero in stringa formattandolo con uno zero iniziale se è composto da una sola cifra. La funzione serve per formattare in maniera più leggibile ore, minuti e secondi del mio orologio. Definisce poi una funzione `aggiornaOrologio()` che crea un oggetto `Date` (`now`) che contiene l'ora e la data correnti. Usando vari metodi della classe **Date** estrae dall'oggetto `now` le informazioni su giorno, mese, anno, giorno della settimana, ora, minuti e secondi. Utilizzando la funzione `zeroPrima()` e i due array le informazioni vengono trasformate in un formato leggibile e quindi usate per aggiornare il campo di testo. Chiamando la funzione `setInterval()` (funzione incorporata nel linguaggio) si imposta un timer del computer in modo che la funzione `aggiornaOrologio()` venga eseguita dieci volte al secondo.

7. Selezionare Controllo > Prova filmato.

`Date` mi può servire anche per fare operazioni sulle date. Ad esempio posso chiedermi quanti giorni sono trascorsi dal 23 dicembre 1981 ad oggi.

1. Creare un nuovo documento Flash e salvarlo come `Date_2 fla`.
2. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
//differenza tra due date
var data_1:Date = new Date(1981,23,11);
```

```
var oggi:Date = new Date();  
  
var diff:Number = oggi.getTime() - data_1.getTime();  
var giorni_trascorsi:Number = Math.round(diff/(3600000*24));  
trace(giorni_trascorsi);
```

Il codice crea due oggetti Date. Il primo contiene il 23 dicembre 1981 (i mesi sono numerati a partire da 0 e quindi dicembre vale 11). Il secondo la data di oggi. Usando il metodo `getTime()` si ottiene il numero di millisecondi trascorsi prima o dopo il 1 gennaio 1970 per entrambe le date. E' possibile quindi calcolare facilmente l'intervallo tra le due date in millisecondi e quindi confertirlo nell'unità di misura che preferisco.

3. Selezionare Controllo > Prova filmato. Sul pannello di output appare il numero di giorni trascorsi dal 23 dicembre 1981.

Creazione di classi

Classi e pacchetti

Nella programmazione orientata agli oggetti una classe definisce una categoria di oggetto. Come abbiamo visto le classi in pratica sono nuovi tipi di dati che il programmatore può creare per definire un nuovo tipo di oggetto. Una classe descrive le proprietà (*dati*) e i metodi (*comportamenti*) 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 (funzioni definite all'interno di una classe) di una classe sono detti membri della classe. In generale per utilizzare le proprietà e i metodi definiti da una classe, è necessario creare prima un'istanza di tale classe. La relazione tra un'istanza e la relativa classe è simile a quella che intercorre tra un edificio e il relativo progetto.

La classe in ActionScript viene sempre definita in un file di script esterno (un normale file di testo con estensione .as) che ha lo stesso nome della classe e che viene chiamato file di classe. Quando un filmato flash viene compilato (utilizzando Controllo > Prova filmato o File > Pubblica) per generare il file .swf, il codice contenuto nei file di classe necessari viene compilato e aggiunto al file .swf.

I pacchetti (packages) 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.

L'uso dei pacchetti serve esclusivamente ad organizzare il meglio il mio lavoro. Nella mia vita di programmatore svilupperò sempre nuovi progetti e presumibilmente creerò nuove classi, che per loro natura sono codice riciclabile, utilizzabile cioè in nuovi progetti. Se organizzerò queste classi in gruppi sarà più facile per me ritrovarle e riutilizzarle.

Per identificare in una applicazione una classe contenuto in un pacchetto utilizzerò la *sintassi del punto*. Nell'esempio precedente la classe WoodWidget la dovrò identificare come Widget.WoodWidget.

L'istruzione import

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` (uno dei pacchetti di classi rilasciato con flash 8) è una directory sul disco rigido che contiene numerosi file di classe, uno per ogni tipo di filtro (`BevelFilter`, `BlurFilter`, `DropShadowFilter` e così via).

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 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 `import` 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 file di classe) 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`).

Creazione di classi personalizzate

Passeremo ora ad esaminare come si costruisce un file di classe.

Partiremo dalla costruzione di una classe molto semplice. Nell'esempio seguente è illustrata l'organizzazione di una classe denominata `User`.

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). Vediamo di definire le regole (alcune obbligatorie, altre solo consigliate) che devo seguire nello scrivere un file di classe saranno presenti questi elementi.

- Il file di classe inizierà con commenti di documentazione, tra cui una descrizione generale del codice e informazioni sull'autore e sulla versione.
- Seguiranno le istruzioni `import` (se necessarie).
- Scriverò poi la dichiarazione di classe come segue:

```
class User {
    ...
}
```

- Inserirò gli eventuali commenti relativi all'implementazione della classe.
- Dichiarerò, quindi, le variabili statiche.

- Dichiarerò le variabili di istanza rispettando questo ordine: prima le variabili pubbliche, poi quelle private.
- Aggiungerò l'istruzione relativa alla funzione di costruzione, come indicato nell'esempio seguente:

```
public function User(username:String, password:String)
{
    ...
}
```

- Scriverò i metodi, raggruppandoli per funzionalità. Questo tipo di organizzazione dei metodi consente di migliorare la leggibilità e la chiarezza del codice.
- Scriverò i metodi getter/setter.

L'esempio seguente illustra una classe ActionScript semplice denominata User.

Per creare il file di classe:

4. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
5. Selezionare File > Salva con nome e assegnare al file il nome User.as.
6. 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;

    // 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;
}
}

```

7. 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.

8. Selezionare File > Nuovo, quindi Documento Flash.
9. Selezionare File > Salva con nome e assegnare al file il nome user_test fla. Salvare il file nella stessa directory di User.as.
10. 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 = "1nu";
user1.password = "1wp";
trace("Dopo:");
trace("\t username = " + user1.username); // 1nu
trace("\t password = " + user1.password); // 1wp

```

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.

11. Salvare il file FLA e selezionare Controllo > Prova filmato per provare i file.

Utilizzo delle classi in una applicazione

Abbiamo visto come è strutturato un file di classe e quali sono le regole per scrivere nei file di classe codice chiaro e leggibile. Nelle sezioni seguenti vedremo come i file di classe vengono utilizzati in un'applicazione. Prima di iniziare riassumiamo i passi in cui si articola la creazione di classi:

1. Definizione di una classe in un file di classe ActionScript esterno.
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.
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.

Nella cartella Esempi > XML_menu potete trovare un file di esempio che dimostra come creare un menu dinamico con dati XML utilizzando una classe appositamente sviluppata. Nel filmato principale (xmlMenu.fla) è presente una chiamata alla funzione di costruzione XmlMenu(), alla quale vengono passati due parametri: il percorso al file di menu XML che contiene i dati che descrivono il menu e il riferimento alla movie clip che ospita il menu. Il resto del codice risiede nel file di classe XmlMenu.as.

Riferimenti alle classi

Abbiamo visto che il codice delle classi viene collocato in un file esterno che viene unito al file swf quando pubblico (cioè compilo) il mio filmato.

Condizione essenziale perché il mio lavoro funzioni è quindi che il compilatore trovi i file di classe necessari.

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.

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 Categoria 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 (sando 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");
// Errore !
//Istruzione non consentita in una definizione di classe.
```

Impostazione e la modifica del 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 sarà 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 (packages) e consentono di organizzare le classi.

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. Tutte i pacchetti di classi fornite con Flash 8 sono, ad esempio, nel percorso di classe globale. 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).

Per default il percorso di classe impostato a livello di documento è vuoto. Quando il documento viene salvato, come percorso di classe viene impostato il percorso in cui il file FLA è stato salvato.

In alcuni 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

Uso di un file di classe in Flash

Per creare un'istanza di una classe ActionScript, si utilizza 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 vuole utilizzare la classe `User` per creare un nuovo oggetto `User`, occorre scrivere:

```
var firstUser:User = new User();
```

`firstUser` conterrà ora un'istanza della classe `User` e a `firstUser` potremo applicare i metodi e leggere o modificare le proprietà definite dalla classe.

Usando l'operatore punto (`.`) si accede al valore di una proprietà di 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
```

Con la stessa sintassi si usano anche 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
```

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. I membri di classe, invece, vengono creati una sola volta per ogni classe. (I membri di una classe sono detti anche membri statici).

Le proprietà 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.

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:

1. Selezionare **File > Nuovo**, quindi selezionare **File ActionScript**.
Salvare il documento come `Picture.as` e quindi fare clic su **OK**.
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/imagel.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.

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 e privati

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, la parola chiave `public` viene utilizzata principalmente per ragioni di stile e leggibilità.

La classe `Sample` seguente contiene un metodo pubblico denominato `myMethod()` e una proprietà privata chiamata `ID`:

```
class Sample {
  private var ID:Number;
  public function Sample(id) {
    this.ID = id;
  }
  public function myMethod():Void {
    //il metodo può modificare la proprietà privata
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Se si desidera aggiungere una proprietà di tipo `public`, si può utilizzare la parola `"public"` anziché `"private"`, come nell'esempio seguente:

```
class Sample {
  private var ID:Number;
  public var email:String;
  public function Sample(id) {
    this.ID = id;
  }
  public function myMethod():Void {
    //il metodo può modificare la proprietà privata
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Come abbiamo detto la direttiva `public` è opzionale (in quanto proprietà e metodi sono pubblici per default), ma si consiglia di utilizzarla per migliorare la leggibilità dello script.

Ora in `Sample` avremo la proprietà pubblica `email` e la proprietà privata `ID`. Se creo un'istanza della classe *vedrò* solo la proprietà `email`.

```
var my_class:Sample = new Sample(34);
my_class.email = pippo@qualcheserver.com; //ok
trace (my_class.email); // ok
trace (my_class.ID); // Errore!!!
```

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..

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 sia pubblici che privati.

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, si fa riferimento al nome della classe anziché a una sua istanza specifica, come nel codice seguente:

```
trace(Math.PI / 8); // 0,392699081698724
```

Nell'esempio della classe `Sample` precedente è possibile creare una variabile (proprietà) statica per tenere traccia del numero di istanze della classe create:

```
class Sample {
    public static var count:Number = 0;
    private var ID:Number;
    public var email:String;
    public function Sample(id) {
        this.ID = id;
        Sample.count++;
        trace("contatore aggiornato: " + Sample.count);
    }
    public function myMethod():Void {
        //il metodo può modificare la proprietà privata
        this.ID = 15;
        trace(this.ID); // 15
        trace("myMethod");
    }
}
```

Ogni volta che viene creata una nuova istanza della classe `Sample`, la 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 utilizzando un'istanza della classe, ma riferendosi alla classe.

La classe di primo livello `Math` è costituita, ad esempio, 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
```

Metodi getter e setter

I metodi getter e setter sono un modo alternativo per definire una proprietà. Quando creo una proprietà definendo una variabile pubblica all'interno della classe consento a chi usa la classe di interagire direttamente con la variabile. Quando leggerà la proprietà leggerà il contenuto della variabile, quando la modificherà il contenuto della variabile. Scrivere metodi setter e getter consente di controllare quanto viene scritto (o letto) ed eventualmente modificarlo. Questa tecnica consente inoltre di creare proprietà a sola lettura e a sola scrittura.

È buona norma, nella programmazione orientata agli oggetti non consentire l'accesso diretto alle proprietà all'interno di una classe. Sarebbe quindi consigliabile creare il maggior numero possibile di variabili di istanza private nelle classi e aggiungere metodi getter e setter per modificarne il contenuto. Bisogna però tener conto di vantaggi e svantaggi e scegliere il proprio stile di lavoro anche sulla base delle proprie inclinazioni. Gestire tutte le proprietà pubblica attraverso metodi setter e getter allunga molto il codice e, se non è corredato da sufficienti commenti, lo rende anche meno leggibile. Dall'altro, però il codice così impostato è più flessibile e cambiare il comportamento di una proprietà risulta più semplice.

Potremmo dire che se lavoriamo a progetti semplici e lavoriamo da soli, probabilmente converrà privilegiare la velocità di scrittura (che significa

anche minore possibilità di errori), se lavoriamo a progetti complessi o lavoriamo in team converrà privilegiare la strutturazione del codice.

In molti casi, comunque, l'uso dei metodi getter e setter è fortemente consigliato o obbligatorio.

Torniamo ad uno degli esempi creati quando abbiamo parlato di metodi statici. Se, ad esempio, nella classe è presente un metodo statico privato che tiene traccia del numero di istanze create per una classe specifica, per evitare errori un utente non deve essere in grado di modificare il contatore tramite codice, ma solo leggerlo.

In una situazione di questo tipo bisogna creare una variabile di istanza privata (che viene incrementata dalla funzione di costruzione della classe) e un metodo getter che restituisce il valore del contatore. In questo modo gli utenti saranno in grado di recuperare il valore corrente solo tramite il metodo getter e non potranno modificarlo. La creazione di un getter senza un setter è un metodo semplice per rendere una proprietà 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 una proprietà denominata `userName`:

```
public var userName:String;
```

Ipotizziamo che io non voglia consentire alle istanze della classe di accedere direttamente a questa proprietà ad esempio

```
user.userName = "Buster";
```

ma di consentire, ad esempio, l'accesso solo in lettura.

Per farlo devo definire la proprietà utilizzando la sintassi dei metodi getter e setter impliciti che consentono di accedere alle proprietà come se fossero pubbliche, ma attraverso un controllo..

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:

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 get userName():String {
        return this.__username;
    }
    public function set userName(value:String):Void {
        this.__username = value;
    }
}
```

3. Salvare il documento ActionScript come Login.as.

Un metodo getter non accetta alcun parametro. Un metodo setter deve accettare un solo parametro obbligatorio. 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 login_test fla nella stessa directory del file Login2.as.
5. Aggiungere il codice ActionScript seguente al fotogramma 1 della linea temporale principale:

```
var user:Login = new Login("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
```

6. Salvare il documento Flash e selezionare Controllo > Prova filmato per provare il file.

Se voglio che la proprietà `userName` sia a sola lettura basta eliminare il comando che imposta la proprietà nel metodo `setter`:

```
class Login {
    private var __username:String;
    public function Login(username:String) {
        this.__username = username;
    }
    public function get userName():String {
        return this.__username;
    }
    public function set userName(value:String):Void {
        //this.__username = value;
    }
}
```

Aggiungo `//` prima del comando in modo che `"this.__username = value"` diventi un commento. Ora se provo a modificare la proprietà

```
user.userName = "EnriqueI";
```

il valore della proprietà non cambia. In una maniera un po' più elegante potrei usare il metodo `setter` per gestire l'errore e segnalare al programmatore che la proprietà è a sola lettura.

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.

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.

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
```

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 programmatori che implementano le classi personalizzate ad accedere a tali variabili tramite metodi getter e setter si eseguirebbe un incapsulamento. 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 creare una classe `Person` che rispetti le caratteristiche di incapsulamento: si impostano i membri di istanza come privati e si definiscono metodi getter e setter per gestirli:

```
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;
    }
}
```


Esempio: Creazione di una classe

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.

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
}
```

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.

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.

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.

- 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 è possibile.
- Dichiarare le variabili prima di utilizzarle, anche nei 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 una variabile locale che ha lo stesso nome della variabile globale.

- 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;

// 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
}
```

Creazione di file di classe e inserimento in pacchetti

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.

I file di classe sono detti anche file ActionScript (AS) e vengono creati nello strumento di modifica di Flash oppure utilizzando un editor esterno:

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`.

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.

- 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.

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
}
```

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.

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. **NOTA**

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.

Creazione della funzione di costruzione

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.

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 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;
}
```

Nella creazione di funzioni di costruzione, tieni presente che:

- 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
- Una funzione di costruzione non può avere un tipo restituito, ma può essere utilizzata solo con l'operatore `new` e restituisce il riferimento alla nuova istanza della classe creata.

Anche le classi incorporate hanno le proprie funzioni di costruzione che invoco quando creo un oggetto appartenente a quella classe:

```
var day_array:Array = new Array("Dom", "Lun", "Mar", "Mer",
    "Gio", "Ven", "Sab");
var somePerson:Person = new Person("Paolo", 30);
```

Per creare un nuovo oggetto `Array` chiamo la funzione di costruzione `Array()` della classe incorporata di primo livello `Array` con l'operatore `new`. Nella stessa maniera per creare una nuova istanza di `Person` chiamo la funzione di costruzione `Person()` con l'operatore `new`.

Aggiungiamo ora una funzione 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 corsivo, 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 invia un semplice messaggio al 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 corsivo, 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.

Aggiunta di metodi e proprietà

Per creare le proprietà per le classi ClassA e ClassB, bisogna definire delle variabili utilizzando la parola chiave var:

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 corsivo, 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 nomeVariabile:Tipo come nei seguenti esempi:

```
var username:String;
var age:Number;
```

Questo stile di programmazione viene definito tipizzazione forte dei dati. In ActionScript, contrariamente ad altri linguaggi, non è strettamente obbligatorio, però è estremamente utile.

Quando si assegna un tipo a una variabile, 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 semplificando molto il debugging di un'applicazione.

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 [{ }]).

Vediamo ora di aggiungere dei metodi alle nostre due classi:

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 corsivo, 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 corsivo crea un nuovo metodo nella classe che manda un semplice messaggio al 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.
4. Aprire classe.as e modificare il codice alla stessa maniera:

```
class com.macromedia.utils.ClassB {
    static var _className:String;
    function ClassB() {
```



```

        trace("funzione di costruzione di ClassB");
    }
    function doSomething():Void {
        trace("ClassB - doSomething()");
    }
}

```

5. Controllare la sintassi di ClassB.as come è stato fatto per ClassA.as.
6. Prima di continuare, salvare entrambi i file ActionScript.

Le proprietà possono essere inizializzate 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 un dato predefinito che non può essere modificato mentre il programma viene eseguito.

Le costanti di compilazione sono: tutti i letterali (stringhe tra virgolette o tra apici, numeri, valori booleani, letterali Array, letterali Object, null e undefined, oltre alle funzioni di costruzione per le seguenti classi di primo livello: Array, Boolean, Number, Object e String.

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 corsivo, 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.

Controllo dell'accesso dei membri di classi (statici)

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` come per i membri di istanza.

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 convenzione 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 `*/`.

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.

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 corsivo 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.

L'abitudine di inserire commenti significativi all'inizio e durante lo sviluppo di un file di classe è estremamente utile anche se non si lavora in gruppo. Passate due settimane da quando avete scritto del codice (magari copiandolo da esempi trovati on line o su qualche manuale) non vi ricorderete tutto e una nota al punto giusto può farvi risparmiare un sacco di tempo.



Disegniamo un rettangolo

In questo capitolo costruiremo insieme passo a passo una classe Rettangolo le attribuiremo delle proprietà e dei metodi la caricheremo in un filmato Flash e proveremo ad utilizzarla. In sostanza applicheremo quanto appreso nel capitolo precedente in termini astratti ad un caso concreto.

Una classe è una componente software riutilizzabile che risolve un problema in termini di programmazione. Lo sviluppo di una nuova classe di oggetti dovrebbe essere indipendente dall'applicazione in cui verrà utilizzata. Più riuscirò a sviluppare componenti indipendenti dalle applicazioni per cui inizialmente le ho create, più avrò prodotto oggetti riutilizzabili.

Per questo il nostro modo di procedere prescindere, per il momento, dall'applicazione in cui l'oggetto potrà venire utilizzato.

Le fasi di realizzazione di una classe possono essere così riassunte:

- Progettazione: comprende il lavoro da svolgere prima della scrittura del codice. Si tratta di definire il funzionamento del nuovo oggetto che la classe mette disposizione, e quindi stabilire quali proprietà e metodi saranno necessari.
- Scrittura del codice. Si passerà quindi alla fase di scrittura del codice. Dovremo realizzare un file di classe secondo le specifiche che abbiamo visto nel capitolo precedente, preoccupandoci soprattutto della corretta sintassi di quanto scriviamo.
- Test e correzione degli errori. Dovremo poi creare un file FLA di prova in cui testare il corretto funzionamento della classe e correggere gli eventuali errori
- Inserimento in una o più applicazioni. Una volta testato il funzionamento della classe passeremo al suo utilizzo in alcune applicazioni flash.

Progettazione

Il nostro obiettivo è creare una classe che gestisca una figura rettangolare di dimensioni definite in modo che:

- Sia possibile ricavare l'area del rettangolo
- Sia possibile disegnare il rettangolo a una determinata coordinata, in un determinato colore e con una determinata trasparenza.
- Sia possibile cancellare e/o spostare il rettangolo disegnato.
- Sia possibile modificare la rotazione del rettangolo.

Per implementare queste caratteristiche devo definire un set di proprietà e di metodi per le istanze della classe.

Prima di tutto dovrò definire le proprietà che posso impostare quando creo un'istanza della mia classe e che definiscono lo stato iniziale del rettangolo. Per le proprietà definisco anche un valore di default, il valore cioè che assumono le proprietà nell'istanza della classe se non sono impostate.

Nome	Descrizione	Visibilità	Tipo	Default	Note
larghezza	La larghezza del rettangolo	private	Number	0	
altezza	L'altezza del rettangolo	private	Number	0	
parent_mc	Clip filmato a cui appartiene il rettangolo.	private	MovieClip	_root	Se la proprietà non è impostata il rettangolo apparterrà a _root
_color	Colore del rettangolo	private	Number	0	Nero se non impostato
_alpha	Trasparenza del rettangolo	private	Number	100	Nessuna trasparenza se non impostata
_rotation	Rotazione del rettangolo relative al clip filmato parent_mc.	private	Number	0	
_x	Posizione del rettangolo (x)	private	Number	Centro dello schermo	
_y	Posizione del rettangolo (y)	private	Number	Centro dello schermo	

Come vedremo tutte queste proprietà dovranno essere impostate dalla funzione di costruzione della classe che inizierà l'istanza creata.

Alcune proprietà (larghezza, altezza e parent_mc) non potranno più essere modificate. Colore, trasparenza, rotazione e posizione potranno essere

modificate, ma non direttamente. Come vedremo per completare il cambiamento di stato (ad esempio per cambiare il colore del rettangolo) avrò bisogno di ridisegnare il rettangolo sullo schermo.

Per gestire le proprietà `_color`, `_alpha`, `_rotation`, `_x` e `_y` avrò quindi bisogno dei corrispondenti setter e getter. Ugualmente definirò un metodo getter e un metodo setter per ottenere l'area.

Nome	Descrizione	Visibilità	Tipo	Default	Note
get color	Restituisce il valore di <code>_color</code>	public	Number	-	
set color (c)	Imposta il valore di <code>_color</code> e ridisegna il rettangolo	public	Number	-	
get alpha	Restituisce il valore di <code>_alpha</code>	public	Number	-	
set alpha (a)	Imposta il valore di <code>_alpha</code> e ridisegna il rettangolo	public	Number	-	
get rotation	La rotazione del rettangolo corrisponde alla rotazione della MovieClip su cui è disegnato.	public	Number	-	
set rotation (r)		public	Number	-	
set x (xpos)	La posizione del rettangolo corrisponde alla posizione della MovieClip su cui è disegnato.	public	Number	-	
get x		public	Number	-	
set y (ypos)		public	Number	-	
get y		public	Number	-	
set area ()	Segnala un errore: proprietà a sola lettura	public	-	-	
get area	Calcola e restituisce l'area del rettangolo	public	Number	-	

Infine `canvas_mc` sarà la MovieClip, creata quando creerò l'istanza della mia classe, a cui applicherò i metodi di disegno per disegnare il rettangolo:

Nome	Descrizione	Visibilità	Tipo	Default	Note
canvas_mc	Clip filmato su cui viene disegnato il rettangolo	private	MovieClip	_root	Viene creata quando viene creata un'istanza della classe

Infine devo definire i metodi per gestire il mio rettangolo: un metodo che ridisegna il rettangolo secondo le impostazioni correnti di colore, trasparenza e rotazione, un metodo che mi consenta di spostare il rettangolo in un determinato punto dello schermo, un metodo che mi consenta di cancellarlo o nascondere e un metodo che mi consenta di eliminarlo definitivamente liberando le risorse create per disegnarlo.

Nome	Descrizione	Visibilità	Parametri	Note
moveTo(xpos,ypos)	Sposta il rettangolo in una determinata posizione (relativa al clip filmato parent_mc)	public	xpos:Number ypos:Number	Le nuove coordinate vengono memorizzate in x e y
draw()	Ridisegna il rettangolo secondo le impostazioni correnti.	public	Nessuno	
clear()	Cancella il rettangolo	public	Nessuno	Il rettangolo viene cancellato ma non eliminato.
free()	Elimina canvas_mc	public	Nessuno	Viene eliminata canvas_mc (necessario per liberare le risorse impiegate creando il rettangolo).

Scrittura del codice

Passiamo ora alla scrittura del codice che tradurrà in un file di classe quanto abbiamo progettato.

Per creare il file di classe:

5. Selezionare File > Nuovo, selezionare File ActionScript e quindi fare clic su OK.
6. Selezionare File > Salva con nome e assegnare al file il nome Rettangolo.as. Salvare il file in una cartella Rettangolo che avremo precedentemente creato.
7. Immettere il codice ActionScript seguente nella finestra Script:

```
/**
Classe Rettangolo
autore: Bruno Migliaretti
versione: 1.0
data modifica: 01/12/2006
copyright: Accademia di Belle Arti di Urbino
Questo codice definisce una classe Rettangolo che consente
di gestire sullo schermo una figura geometrica di forma
rettagolare.
*/

class Rettangolo {
}
```

8. Salvare il file.

NOTA

E' molto importante che il nome del file di classe (esclusa l'estensione as) corrisponda esattamente (compreso l'uso di maiuscole e minuscole) al nome della classe creata. Se il nome non dovesse corrispondere, il compilatore non sarebbe in grado di trovare il codice di classe quanto userò la classe in un file fla.

Ora dovrò dichiarare tutte le proprietà della classe, prima le proprietà private poi quelle pubbliche.

9. Inserire il codice in corsivo nello script:

```
/**
  Classe Rettangolo
  autore: Bruno Migliaretti
  versione: 1.0
  data modifica: 01/12/2006
  copyright: Accademia di Belle Arti di Urbino
  Questo codice definisce una classe Rettangolo che consente
  di gestire sullo schermo una figura geometrica di forma
  rettangolare.
*/

class Rettangolo {
  private var larghezza:Number;           //La larghezza del
                                           //rettangolo

  private var altezza:Number;             //L'altezza del
                                           //rettangolo
  private var parent_mc:MovieClip;         //Moviclip a cui
                                           //appartiene il
                                           //rettangolo

  private var _color:Number;               //Colore del
                                           //rettangolo
  private var _alpha:Number;               //Trasparenza del
                                           //rettangolo
  private var _rotation:Number;            //Rotazione del
                                           //rettangolo.

  private var _x:Number;                   //Coordinata x
  private var _y:Number;                   //Coordinata y
  private var canvas:MovieClip;            //MovieClip su cui
                                           //viene disegnato
                                           //il rettangolo
}
```

10. Prima di proseguire salvare il file.

A questo punto occorre scrivere la funzioni di costruzione che ha lo stesso nome della classe. Quando un'istanza di classe viene creata saranno impostate le proprietà che determinano l'aspetto del rettangolo. La funzione di costruzione accetta fino a sei parametri: larghezza, altezza, clip filmato a

cui appartiene il rettangolo, colore, trasparenza, rotazione. La funzione controlla il contenuto dei parametri e se il loro valore è diverso da undefined imposta le relative proprietà ai valori dei parametri, altrimenti le imposta ai valori di default.

11. Inserire il codice in corsivo nello script:

```
/**
  Classe Rettangolo
  autore: Bruno Migliaretti
  versione: 1.0
  data modifica: 01/12/2006
  copyright: Accademia di Belle Arti di Urbino
  Questo codice definisce una classe Rettangolo che consente
  di gestire sullo schermo una figura geometrica di forma
  rettangolare.
*/

class Rettangolo {
  private var larghezza:Number;           //La larghezza del
                                           //rettangolo

  private var altezza:Number;             //L'altezza del
                                           //rettangolo
  private var parent_mc:MovieClip;         //Moviclip a cui
                                           //appartiene il
                                           //rettangolo

  private var _color:Number;               //Colore del
                                           //rettangolo
  private var _alpha:Number;               //Trasparenza del
                                           //rettangolo
  private var _rotation:Number;            //Rotazione del
                                           //rettangolo.

  private var x:Number;                    //Coordinata x
  private var y:Number;                    //Coordinata y
  private var canvas:MovieClip;            //MovieClip su cui
                                           //viene disegnato
                                           //il rettangolo

  public function Rettangolo(larg:Number, alt:Number,
                             parent:MovieClip, col:Number,
                             trans:Number, rot:Number)
  {
    // vengono impostate le proprietà
    if (larg != undefined) {
      this.larghezza = larg;
    } else {
      this.larghezza = 0;
    }
    if (alt != undefined) {
      this.altezza = alt;
    } else {
      this.altezza = 0;
    }
    if (parent_mc != undefined) {
```

```

        this.parent_mc = parent;
    } else {
        this.parent_mc = _root; //timeline principale
    }
    if (col != undefined) {
        this._color = col;
    } else {
        this._color = 0;          //nero
    }
    if (trans != undefined) {
        this._alpha = trans;
    } else {
        this._alpha = 100;       //non trasparente
    }
    if (rot != undefined) {
        this._rotation = rot;
    } else {
        this._rotation = 0;      //in gradi
    }

    // viene creata la movie clip su cui verrà disegnato
    // il rettangolo
    canvas_mc = parent_mc.createEmptyMovieClip("canvas_mc",
        parent_mc.getNextHighestDepth());

    //il rettangolo per default viene collocato al centro
    //dello schermo
    this.moveTo(Stage.width/2, Stage.height/2);
}
}

```

12. Prima di proseguire salvare il file.

La MovieClip canvas_mc è il mezzo che ci consente di disegnare il rettangolo autonomamente dagli altri oggetti eventualmente sullo stage. Il rettangolo disegnato diventa di fatto la MovieClip canvas_mc. Quando creo una nuova istanza di Rettangolo devo quindi creare una nuova MovieClip vuota su cui disegnare con i metodi che la classe MovieClip mi mette a disposizione. Per farlo devo applicare alla clip filmato madre (parent_mc) il metodo createEmptyMovieClip che appunta crea una clip vuota in parent_mc. Il metodo accetta due parametri il nome della nuova clip creata (nel nostro caso "canvas_mc") e la *profondità* della clip. Quando esistono più oggetti su una movie clip ogni oggetto ha la sua profondità (depth) che determina se viene mostrato dietro o davanti ad un altro oggetto figlio della stessa clip. Il metodo getNextHighestDepth mi restituisce appunto il livello di profondità in primo piano. In questo modo la clip creata risulterà davanti a tutti gli altri oggetti presenti in quel momento nella clip madre.

Ora aggiungo alla mia classe i metodi draw() e moveTo(xpos,ypos).

13. Inserire il codice in corsivo nello script:

```
/**
```

```

Classe Rettangolo
// ...codice omissso per brevità
*/

class Rettangolo {
    // ...codice omissso per brevità
    public function Rettangolo(larg:Number, alt:Number,
                                parent:MovieClip, col:Number,
                                trans:Number, rot:Number)

    {
        // ...codice omissso per brevità
    }

    public function draw ():Void {
        /* La MovieClip canvas_mc viene svuotata (clear) e viene
        disegnato un rettangolo nel colore e con la
        trasparenza definiti dalle proprietà color e alpha.
        Il rettangolo viene disegnato in modo che le
        coordinate di l'origine di canvas_mc risultino
        al centro del rettangolo.
        */
        canvas_mc.clear();
        canvas_mc.lineStyle(0,0,0);
        canvas_mc.beginFill(this.color, this.alpha);
        canvas_mc.moveTo(0 - this.larghezza/2 ,
                        0 - this.altezza/2);
        canvas_mc.lineTo(this.larghezza/2, 0 - this.altezza/2);
        canvas_mc.lineTo(this.larghezza/2, altezza/2);
        canvas_mc.lineTo(0 - this.larghezza/2, altezza/2);
        canvas_mc.lineTo(0 - this.larghezza/2,this.altezza/2);
        canvas_mc.endFill();
        canvas_mc._rotation = this.rotation
    }

    public function moveTo(xpos:Number, ypos:Number) {
        // aggiorno le coordinate del rettangolo
        this.x = xpos;
        this.y = ypos;
    }
}

```

14. Prima di proseguire salvare il file.

Il metodo draw() disegna il rettangolo nel colore e secondo la trasparenza stabiliti sulla clip canvas_mc. Vengono utilizzati i metodi di disegno forniti dalla classe MovieClip. Ulteriori informazione si possono trovare sulla Guida di Riferimento ad ActionScript. Il rettangolo viene disegnato in modo che la coordinata d'origine (0,0) di canvas_mc risulti al centro del rettangolo. Per ottenere la rotazione del rettangolo si agisce ruotando canvas_mc.

Il metodo moveTo() non fa che aggiornare le coordinate (le proprietà x e y) del rettangolo sarà il relativo metodo setter (che ancora devo scrivere) a gestire al modifica di posizione.

Inseriamo ora gli altri metodi che avevamo definito a livello di progettazione:

15. Inserire il codice in corsivo nello script:

```
/**
  Classe Rettangolo
  // ...codice omissso per brevità
*/

class Rettangolo {
  // ...codice omissso per brevità
  public function Rettangolo(larg:Number, alt:Number,
                             parent:MovieClip, col:Number,
                             trans:Number, rot:Number)

  {
    // ...codice omissso per brevità
  }

  public function draw ():Void {
    // ...codice omissso per brevità
  }

  public function moveTo(xpos:Number, ypos:Number):Void {
    // ...codice omissso per brevità
  }

  public function clear ():Void {
    //elimina ogni disegno presente su canvas_mc
    this.canvas_mc.clear();
  }

  public function free ():Void {
    //elimina canvas_mc liberando risorse
    canvas_mc.removeMovieClip();
  }
}
```

16. Prima di proseguire salvare il file.

Ora non resta che aggiungere i metodi getter e setter per la gestione della posizione, del colore, della trasparenza della rotazione e dell'area.

17. Aggiungiamo i metodi che gestiscono la posizione:

```
class Rettangolo {
  // ...codice omissso per brevità

  //la posizione del rettangolo è la posizione di canvas_mc
  public function get x ():Number {
    return (this.camvas_mc._x);
  }
  public function set x (xpos:Number) {
    this.camvas_mc._x = xpos;
  }
}
```

```

    public function get y ():Number {
        return (this.canvas_mc._y);
    }
    public function set y (ypos:Number) {
        this.canvas_mc._y = ypos;
    }
}

```

18. Aggiungiamo i metodi che gestiscono il colore:

```

class Rettangolo {
    // ...codice omissso per brevità

    public function set color (c:Number) {
        //dopo aver modificato la proprietà _color
        //viene chiamato draw per aggiornare la variazione
        //sullo schermo
        this._color = c;
        this.draw();
    }
    public function get color ():Number {
        return this._color;
    }
}

```

19. Aggiungiamo i metodi che gestiscono la trasparenza:

```

class Rettangolo {
    // ...codice omissso per brevità

    public function set alpha (c:Number) {
        //dopo aver modificato la proprietà _alpha
        //viene chiamato il metodo draw per aggiornare
        //la variazione sullo schermo
        this._alpha = c;
        this.draw();
    }
    public function get alpha ():Number {
        return this._alpha;
    }
}

```

20. Aggiungiamo i metodi che gestiscono la rotazione:

```

class Rettangolo {
    // ...codice omissso per brevità

    //la rotazione del rettangolo è la rotazione di canvas_mc
    public function get rotation ():Number {
        return (this._rotation);
    }
}

```

```

    }
    public function set rotation (r:Number) {
        this._rotation = r;
        this.canvas_mc._rotation = r;
    }
}

```

21. Infine l'area:

```

class Rettangolo {
    // ...codice omissso per brevità

    //calcolo dell'area
    public function get area ():Number {
        return (this.larghezza * this.altezza);
    }
    public function set area (a:Number) {
        //manda alla finestra di output un messaggio di errore
        trace("Errore: proprietà a solo scrittura");
    }
}

```

La funzione **set area()** si limita a mandare un messaggio di errore alla finestra di output per avvertire il programmatore che sta tentando di impostare una proprietà a sola lettura.

Alla fine il file ottenuto conterrà questo codice:

```

/**
 *ClasseRettangolo
 *autore: Bruno Milgiaretti
 *versione: 1.0
 *data modifica: 01/12/2006
 *copyright: Accademia di Belle Arti di Urbino
 *Questo codice definisce una classe Rettangolo che consente di gestire sullo schermo
 *una figura geometrica di forma rettangolare.
 */

class Rettangolo {
    private var larghezza:Number;           //La larghezza del rettangolo
    private var altezza:Number;             //L'altezza del rettangolo
    private var parent_mc:MovieClip;        //MovieClip a cui appartiene il rettangolo

    private var _color:Number;              //Colore del rettangolo
    private var _alpha:Number;              //Trasparenza del rettangolo
    private var _rotation:Number;           //Rotazione del rettangolo relativa al clip
    filmato parent_mc.

    private var _x:Number;                  //Coordinata x del
    rettangolo relativa al clip filmato parent_mc
    private var _y:Number;                  //Coordinata y del
    rettangolo relativa al clip filmato parent_mc
    private var canvas_mc:MovieClip;        //MovieClip su cui è disegnato il
    rettangolo

    public function Rettangolo (larg:Number, alt:Number, parent:MovieClip, col:Number,
                                trans:Number, rot:Number) {

        //vengono impostate le proprietà
        if (larg != undefined) {
            this.larghezza = larg;
        } else {
            this.larghezza = 0;
        }
        if (alt != undefined) {
            this.altezza = alt;
        } else {

```

```

        this.altezza = 0;
    }
    if (parent_mc != undefined) {
        this.parent_mc = parent;
    } else {
        this.parent_mc = _root;           //timeline principale
    }
    if (col != undefined) {
        this._color = col;
    } else {
        this._color = 0;                  //nero
    }
    if (trans != undefined) {
        this._alpha = trans;
    } else {
        this._alpha = 100;                //non trasparente
    }
    if (rot != undefined) {
        this._rotation = rot;
    } else {
        this._rotation = 0;                //in gradi
    }

    // viene creata la movie clip su cui verrà disegnato il rettangolo
    this.canvas_mc =
    parent_mc.createEmptyMovieClip("canvas",parent_mc.getNextHighestDepth());

    //il rettangolo per default viene collocato al centro dello schermo
    this.moveTo(Stage.width/2, Stage.height/2);

}

public function draw ():Void {
    /*
    La MovieClip canvas_mc viene svuotata (clear) e viene disegnato un rettangolo nel
    colore e con la
    trasparenza definiti dalle proprietà color e alpha. Il rettangolo viene disegnato
    in modo che le
    coordinate di l'origine di canvas_mc risultino al centro del rettangolo.
    */
    this.canvas_mc.clear();
    this.canvas_mc.lineStyle(0,0,0);
    this.canvas_mc.beginFill(this.color, this.alpha);
    this.canvas_mc.moveTo(0 - this.larghezza/2 ,0 - this.altezza/2);
    this.canvas_mc.lineTo(this.larghezza/2, 0 - this.altezza/2);
    this.canvas_mc.lineTo(this.larghezza/2, altezza/2);
    this.canvas_mc.lineTo(0 - this.larghezza/2, altezza/2);
    this.canvas_mc.lineTo(0 - this.larghezza/2,this.altezza/2);
    this.canvas_mc.endFill();
    this.canvas_mc._rotation = this._rotation
}

public function moveTo(xpos:Number, ypos:Number):Void {
    // aggiorni le coordinate del rettangolo
    this.x = xpos;
    this.y = ypos;
}

public function clear ():Void {
    //elimina ogni disegno presente su canvas_mc
    this.canvas_mc.clear();
}

public function free ():Void {
    //elimina canvas_mc liberando risorse
    canvas_mc.removeMovieClip();
}

//la posizione del rettangolo è la posizione di canvas_mc
public function get x ():Number {
    return (this.canvas_mc._x);
}
public function set x (xpos:Number) {
    this.canvas_mc._x = xpos;
}

public function get y ():Number {
    return (this.canvas_mc._y);
}
public function set y (ypos:Number) {
    this.canvas_mc._y = ypos;
}

```

```

public function set color (c:Number) {
//dopo aver modificato la proprietà _color
//viene chiamato il metodo draw per aggiornare
//la variazione sullo schermo
this._color = c;
this.draw();
}
public function get color ():Number {
return this._color;
}

public function set alpha (c:Number) {
//dopo aver modificato la proprietà _alpha
//viene chiamato il metodo draw per aggiornare
//la variazione sullo schermo
this._alpha = c;
this.draw();
}
public function get alpha ():Number {
return this._alpha;
}

//la rotazione del rettangolo è la rotazione di canvas_mc
public function get rotation ():Number {
return (this._rotation);
}
public function set rotation (r:Number) {
this._rotation = r;
this.canvas_mc._rotation = r;
}

//calcolo dell'area
public function get area ():Number {
return (this.larghezza * this.altezza);
}
public function set area (a:Number) {
//manda alla finestra di output un messaggio di errore
trace("Errore: proprietà a solo scrittura");
}
}

```

Il file Rettangolo.as è nella cartella “Esempi\Rettangolo” del CD.

Proviamo la classe Rettangolo

Per provare la classe Rettangolo:

22. Selezionare File > Nuovo, quindi Documento Flash.
23. Selezionare File > Salva con nome e assegnare al file il nome rettangolo_prova.fla. Salvare il file nella stessa directory dove si trova Reattangolo.as.
24. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```

var r:Rettangolo;
r = new Rettangolo(400,300,this,0xFF0000,50,30);
r.draw();

```

Il codice ActionScript nel documento Flash è molto semplice. La variabile `r` viene dichiarata di tipo `Rettangolo`. Viene creata una nuova istanza della classe `Rettangolo` e viene assegnata alla variabile `r`. Il rettangolo ha una larghezza di 400 pixel, un'altezza di 300 pixel, fa parte della MovieClip corrente (`this`) che nello specifico è il filmato principale (`_root`), verrà disegnato in rosso (`FF0000`), trasparente al 50% e inclinato di 30 gradi.

25. Scegliere Controllo>Prova filmato.

Sullo schermo apparirà un rettangolo rosso parzialmente trasparente.

26. Aggiungere il codice ActionScript seguente allo script:

```
var r:Rettangolo;  
r = new Rettangolo(400,300,this,0xFF0000,50,30);  
r.draw();  
trace("Il rettangolo è composta da " + r.area + " pixel.");
```

Nella finestra di output apparirà: *Il rettangolo è composta da 120000 pixel.*

Nei file rettangolo_prova_2.fla, rettangolo_prova_3.fla, rettangolo_prova_4.fla, potrete trovare altri esempi di utilizzo della classe Rettangolo in semplici applicazioni.

L'ereditarietà

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*. Posso creare una classe estendendo una classe predefinita. Se creo una classe personalizzata che estende la classe `MovieClip`, ad esempio, la sottoclasse *eredita* tutte le proprietà e i metodi della classe `MovieClip`. Ma posso anche creare un set di classi che estenda una superclasse sempre creata da me. La classe `Pomodoro`, la classe `Melanzana` e la classe `Sedano`, ad esempio, potrebbe essere create estendendo la superclasse `Verdura`.

La sottoclasse può aggiungere dei metodi alla superclasse e ridefinire i metodi ereditati creando così nuovi comportamenti per gli stessi metodi.

L'uso di sottoclassi consente di riutilizzare il codice, in quanto estendendo una classe esistente si evita di riscrivere tutto il codice comune a entrambe le classi. 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. Potrei, 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`. Potrei poi aggiungere una nuova classe `Manager` che estenda a sua volta la classe `Dipendente` e definisca metodi e proprietà aggiuntivi come `assumere()`, `licenziare()`, `aumentare()` e `promuovere()`. Potrei infine estendere la sottoclasse `Manager`, e creare una nuova classe `Direttore` che aggiunga nuovi metodi o sostituisca i metodi esistenti.

Senza l'ereditarietà necessario riscrivere ogni metodo e proprietà in ogni file di classe, anche se le funzionalità sono le stesse nelle classi simili, perdendo tempo nella scrittura del codice e rischiando di aggiungere errori.

In ActionScript, per determinare l'ereditarietà tra una classe e la relativa superclasse o per estendere un'interfaccia viene utilizzata la parola chiave `extends`.

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 le classi personalizzate e molte delle classi di base e delle classi ActionScript di Flash Player. Non possono essere estese la classe `TextField` e le classi statiche, quali `Math`, `Key` e `Mouse`.

Per creare questo tipo di relazione tra due classi, è necessario utilizzare la clausola `extends` dell'istruzione `class`:

```
class SubClass extends SuperClass {
    //corpo della classe
}
```

La classe specificata come `SubClass` eredita tutte le proprietà e i metodi definiti in `SuperClass`.

Aggiunta di metodi

Ora definiremo una classe personalizzata `JukeBox` che estende la classe `Sound`: vengono aggiunti la proprietà `song_array` di tipo `Array` e la proprietà `titles_array` sempre di tipo `Array`, il metodo `playSong()` che riproduce un brano musicale richiamando il metodo `loadSound()` ereditato dalla classe `Sound`, il metodo `addSong()` che aggiunge un brano al `JukeBox` e il metodo `getTitleList()` che restituisce la lista dei titoli inseriti nel `JukeBox`.

Per creare la classe `JukeBox` procedere così:

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 `JukeBox.as`. Salvare il file in una cartella `JukeBox` che avremo precedentemente creato.
3. Immettere nella finestra `Script` il codice ActionScript che documenta la classe `JukeBox` e l'istruzione `class` per creare la classe. La clausola `extends Sound` comunica al compilatore che la classe `JukeBox` è un'estensione della classe `Sound`:

```
/**
 * Classe JukeBox
 * autore: Bruno Migliaretti
 * versione: 1.0
 */
```

```

data modifica: 01/12/2006
copyright: Accademia di Belle Arti di Urbino
Questo codice definisce una classe JukeBox che estende la
classe prdefinita Sound e consente di gestire una lista di
brani musicali.
*/

class JukeBox extends Sound {
}

```

4. Salvare il file.
5. Definire e inizializzare ora nel codice le proprietà song_array e titles_array aggiungendo al corpo della classe:

```

class JukeBox extends Sound {
  private var song_array:Array = new Array();
    //lista dei brani del JukeBox
  private var titles_array:Array = new Array();
    //lista dei titoli del JukeBox
}

```

6. Salvare il file.
7. Aggiungere ora il codice per il metodo addSong():

```

public function addSong (titolo:String,
                        nomeFile:String):Number {
    //aggiungo in coda ai due array il titolo del brano
    //e il file musicale corrispondente
    this.titles_array.push(titolo);
    this.song_array.push(nomeFile);

    //restituisco l'indice del brano inserito
    return (song_array.length - 1);
}

```

Il metodo addSong() accetta due parametri entrambi di tipo String: *titolo* e *nomeFile*. I due elementi vengono aggiunti rispettivamente agli array song_array e titles_array. Infine il metodo restituisce l'indice dell'elemento appena inserito.

8. Salvare il file.
9. Aggiungere ora il codice per il metodo playSong()

```

public function playSong (songId:Number) {
    //il file memorizzato nella proprietà url dell'oggetto
    //con indice songId viene caricato ed eseguito
    this.loadSound(song_array[songId], true);
}

```

10. Aggiungere ora il codice per il metodo getTitleList()

```

public function getTitleList():Array {
    //in questo modo titles_array è accessibile solo in
    lettura e
    //può essere modificato solo usando il metodo addSong
    return titles_array;
}

```

Il metodo getTitleList() consente l'accesso in lettura alla proprietà title_array che deve poter essere modificata solo dal metodo addSong() e quindi è definita come privata.

11. Salvare il file.

Il codice dovrebbe ora risultare:

```

/**
 * Classe JukeBox
 * autore: Bruno Migliaretti
 * versione: 1.0
 * data modifica: 01/12/2006
 * copyright: Accademia di Belle Arti di Urbino
 * Questo codice definisce una classe JukeBox che estende la classe predefinita Sound
 * e consente di gestire una lista di brani musicali.
 */
class JukeBox extends Sound {
    private var song_array:Array = new Array(); //lista dei brani del JukeBox
    private var titles_array:Array = new Array(); //lista dei titoli del JukeBox
    public function addSong (titolo:String, nomeFile:String):Number {
        //aggiungo in coda ai due array il titolo del brano
        //e il file musicale corrispondente
        this.titles_array.push(titolo);
        this.song_array.push(nomeFile);
        //restituisco l'indice del brano inserito
        return (song_array.length - 1);
    }

    public function playSong (songId:Number) {
        //il file memorizzato nella proprietà url dell'oggetto con indice songId
        //viene caricato ed eseguito
        this.loadSound(song_array[songId], true);
    }

    public function getTitleList():Array {
        //in questo modo titles_array è accessibile solo in lettura e
        //può essere modificato solo usando il metodo addSong
        return titles_array;
    }
}

```

Usiamo ora la nostra classe JukeBox in una applicazione. Utilizzando la componente list creeremo un piccolo jukebox interattivo.

12. Selezionare File > Nuovo, quindi Documento Flash.

13. Selezionare File > Salva con nome e assegnare al file il nome jukebox_prova fla. Salvare il file nella stessa directory dove si trova JukeBox.as. Nella cartella \Esempi\JukeBox del CD troverete anche alcuni file mp3 utili per provare l'applicazione.

14. Con lo strumento di creazione del testo create un testo statico e scrivete *Lista dei brani*.

15. Collocate poi sullo Stage un'istanza della componente List trascinandone l'icona dalla finestra componenti allo stage. Usando il pannello proprietà date alla componente il nome song_list.
16. Create un nuovo livello e chiamatelo Azioni. Selezionate il primo fotogramma del nuovo livello e immettete il codice ActionScript seguente:

```
//creo un'istanza di JukeBox
var jb:JukeBox = new JukeBox();

// aggiungo le informazioni sui brani musicali
jb.addSong("Blue", "0169_Blue.mp3");
jb.addSong("Cafe caldo", "0217_Cafe Caldo.mp3");
jb.addSong("Cello sweet", "0246_Cello Sweet.mp3");
jb.addSong("Conto alla rovescia", "0328_Countdown.mp3");
jb.addSong("Vai facile", "0459_Easy Does It.mp3");
jb.addSong("Eine Kleine Nachtmusik", "0467_Eine Kleine Nachtmusik.mp3");
jb.addSong("Gymnopedie N.1", "0663_Gymnopedie N.1.mp3");
jb.addSong("Kyrie Osanna", "0843_Kyrie Osanna.mp3");
jb.addSong("Lighthouse", "0903_Lighthouse.mp3");
jb.addSong("Long Kyrie", "0917_Long Kyrie.mp3");
jb.addSong("Morning Atmosphere", "1013_Morning Atmosphere.mp3");
jb.addSong("Playa Vista", "1182_Playa Vista.mp3");
jb.addSong("Il bel Danubio Blu", "1584_The Blue Danube.mp3");
jb.addSong("The Emperor's Breakfast", "1593_The Emperor's Breakfast.mp3");

//riempio la lista song_list
var titoli:Array = jb.getTitleList();

for (var i = 0; i < titoli.length; i++) {
    song_list.addItem({label:titoli[i],data:i});
}

//gestisco l'evento change delle lista
var listHandler = new Object();

listHandler.change = function(evt:Object) {
    var index = evt.target.selectedItem.data;
    jb.playSong(index);
}

this.song_list.addEventListener("change",listHandler);
```

Al posto dei brani contenuti nella cartella \Esempi\JukeBox potete inserire i vostri brani, inserendo per ogni brano il titolo e l'indirizzo del file. L'indirizzo può essere sia un indirizzo relativo (come nell'esempio in cui i file sono nella stessa cartella dell'applicazione, sia un indirizzo assoluto come, ad esempio, una url completa (come [http://www.unserver.org/una cartella/unfile.mp3](http://www.unserver.org/una%20cartella/unfile.mp3)).

Sostituzione di metodi e 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 aggiungendo metodi e proprietà, ma anche di modificare le funzionalità già presenti.

Come esempio modificheremo il comportamento del metodo toString() nella classe predefinita Date creando una classe che chiameremo DataIt.

Il metodo toString() è comune a tutte le classi predefinite e viene invocato automaticamente quando un'istanza di una qualsiasi classe viene usata come una stringa. Facciamo una prova:

1. Selezionare File > Nuovo, quindi Documento Flash.
2. Selezionare File > Salva con nome e assegnare al file il nome datait_prova_1 fla. Salvare il file nella stessa directory dove abbiamo intenzione di salvare il file di classe DataIt.as che creeremo tra poco.
3. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
//creo una nuova istanza di Date per la data e l'ora corrente
var now:Date = new Date();

//la mando all finestra di output
trace(now);
```

Sulla finestra di output verrà scritto:

Tue Nov 28 22:07:32 GMT+0100 2006

Quando passo un oggetto come parametro alla funzione trace() (che manda un messaggio sulla finestra di output) di fatto lo converto in stringa e quindi applico implicitamente il metodo toString() all'oggetto.

Se provo infatti a cambiare il codice sopra in questa maniera:

```
//creo una nuova istanza di Date per la data e l'ora corrente
var now:Date = new Date();

//la mando all finestra di output
trace(now.toString());
```

Il risultato non cambierà.

Modificando quindi il comportamento standard del metodo toString() per un oggetto personalizzerò il modo con cui quell'oggetto verrà convertito in stringa.

Ecco come posso procedere:

1. Create un nuovo documento ActionScript e salvatelo come DataIt.as. Salvatelo nella stessa cartella in cui è stato precedentemente salvato datait_prova_1 fla.

2. In DataIt.as, immettere il codice ActionScript seguente nella finestra Script:

```
/**
 * Classe DataIt
 * autore: Bruno Migliaretti
 * versione: 1.0
 * data modifica: 01/12/2006
 * copyright: Accademia di Belle Arti di Urbino
```

```

Questo codice definisce una classe DataIt che estende la classe predefinita Date
traducendone l'output in italiano.
*/

class DataIt extends Date {
    //creo una proprietà mesi che contiene i nomi dei mesi
    private var mesi:Array = new Array("gennaio", "febbraio", "marzo", "aprile",
                                        "maggio", "giugno", "luglio", "agosto",
                                        "settembre", "ottobre", "novembre", "dicembre");

    //e una proprietà giorni che contiene i nomi dei giorni della settimana
    private var giorni:Array = new Array( "domenica", "lunedì", "martedì", "mercoledì",
                                        "giovedì", "venerdì", "sabato");

    //scrivo una semplice metodo che aggiunge uno "0"
    //davanti ad un numero se è composto da una sola cifra
    private function zeroPrima(n:Number):String {
        var s:String = n.toString();
        if (s.length == 1) {
            s = "0" + s;
        }
        return (s);
    }

    //ridefinisco il metodo toString()
    public function toString():String {
        var giorno_sett:String = giorni[this.getDay()];
        var giorno:Number = this.getDate();
        var mese:String = mesi[this.getMonth()];
        var anno:Number = this.getFullYear();
        var ora:String = zeroPrima(this.getHours());
        var minuti:String = zeroPrima(this.getMinutes());
        var secondi:String = zeroPrima(this.getSeconds());
        return( giorno_sett + " " + giorno + " " + mese + " " + anno +
                " " + ora + ":" + minuti + ":" + secondi);
    }
}

```

4. Tornare a datait_prova_1.fla e modificare il codice ActionScript nel fotogramma 1 della linea temporale nel modo seguente:

```

//creo una nuova istanza di Date per la data e l'ora corrente
var now:Date = new Date();
//la mando all finestra di output
trace("Now = " + now);

//creo una nuova istanza di DataIt per la data e l'ora
//corrente
var adesso:DataIt = new DataIt();
//la mando all finestra di output
trace("Adesso = " + adesso);

```

L'output sulla finestra di output sarà:

```

Now = Tue Nov 28 22:30:40 GMT+0100 2006
Adesso = martedì 28 novembre 2006 22:30:40

```

Inseriamo la nostra classe DataIt in un'applicazione.

1. Creare un nuovo documento Flash e salvarlo come datait_prova_1.fla nella cartella dove abbiamo salvato DataIt.as.
2. Creare sullo stage un campo di testo dinamico. Scegliere come font del campo _sans e dargli il nome messaggio_txt.

3. Immettere il codice ActionScript seguente nel fotogramma 1 della linea temporale:

```
//semplice orologio-calendario perpetuo
//utilizza setInterval() e la classe DataIt

stop();

//definisco la funzione aggiornaOrologio()
// che aggiorna il contenuto del campo di testo messaggio_txt

function aggiornaOrologio ():Void {
    messaggio_txt.text = new DataIt();
}

// utilizzando il timer del computer la funzione
// aggiornaOrologio
// viene eseguita ogni 100 millisecondi
setInterval(aggiornaOrologio, 100);
```

4. Selezionare Controllo > Prova filmato.

Esercizio

Provare a realizzare una classe DateEs che abbia l'output della data in spagnolo derivandola da DataIt e scrivendo il minimo codice necessario.

Appendici

I testi fondamentali per imparare ad utilizzare Flash e ActionScript sono:

- Apprendimento di ActionScript 2.0 in Flash che trovate nella cartella Documentazione\Action Script del cd. Una buona parte di questa dispensa deriva da questo testo. Ho cercato di sintetizzare e, soprattutto, di semplificare e rendere più chiari i concetti fondamentali. In questa appendice allego, per vostra comodità, tre capitoli della guida Adobe, perché fanno parte integrante di un percorso di apprendimento su ActionScript.
- Guida di riferimento di ActionScript 2.0 che trovate nella cartella Documentazione\Action Script del cd. È un testo di consultazione che replica l'help on line e vi fornisce tutti i dettagli sui comandi, le funzioni e le classi predefiniti di ActionScript.
- Uso dei componenti e Guida di riferimento dei componenti che trovate nella cartella Documentazione\Componenti che vi forniscono tutte le informazioni per utilizzare le componenti offerte insieme a Flash 8.
- Uso di Flash e Guida introduttiva di Flash che trovate nella cartella Documentazione\Uso di Flash e che forniscono informazioni soprattutto sull'ambiente di sviluppo orientate ai principianti.

Completando la documentazione una cartella esercitazioni dove troverete il testo Esercitazioni di Flash e alcuni file FLA legati al testo.

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`, `ContextMenuItems`, `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\Flex 8\Samples and Tutorials\Samples\ActionScript\StageSize.
- In Macintosh, scegliere *Macintosh HD*/Applicazioni/Macromedia Flex 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 dei 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.removeListener(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`.

6. 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);
```

7. 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:

1. Trascinare un'istanza del componente `Loader` dal pannello Componenti nello stage.
2. Nella finestra di ispezione Proprietà, selezionare il loader e digitare `my_ldr` nella casella di testo Nome istanza.
3. 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](#)
- [“Specifica 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 [“Specifica 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 virgole. 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_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcListener);
img_mcl.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 **mcbutton 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 **mcbutton** 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`.

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 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

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\Flesh 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 prevede 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 imporre 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 riferiscano 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Animation.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript/Galleries.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Animation.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flesh 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\Flesh 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, deselezionare 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Animation.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 la riproduzione del file SWF, il che consente un rendering più rapido del file.

NOTA

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 <code>scrollRect</code> (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\FashType.

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\Fash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Fash 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\Fash 8\Samples and Tutorials\Samples\ActionScript\FashType.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Fash 8/Samples and Tutorials/Samples/ActionScript/FashType.

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\Flex 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript/FlashType.
- In Macintosh, accedere a Macintosh HD/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
 - In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 scorrimento 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 imposta `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_mc1:MovieClipLoader = new MovieClipLoader();
my_mc1.addListener(mcListener);
my_mc1.loadClip("http://www.helpexamples.com/flash/images/imagel.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 sottoclasse 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 sottoclasse 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 Insieme 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 <code>TextField</code>	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.

NOTA

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\Flex 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*\Applicazioni\Macromedia Flex 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 `textfieldA.fla` e `textfieldB.fla`, si trovano nella cartella `Samples` del disco rigido:

- In Windows, accedere a *unità di avvio*\Programmi\Macromedia\Flex 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Fish 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Fish 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 Fish` 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\Flex 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 Flex 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 Flex 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, Flex 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 `textfieldA fla` e `textfieldB fla`, si trovano nella cartella `Samples` del disco rigido:

- In Windows, accedere a *unità di avvio*\Programmi\Macromedia\Flex 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 Flex, ad esempio `FlashVars`, `LoadVars`, XML o i servizi Web, per citarne alcuni. Il metodo probabilmente più semplice per passare testo a un documento Flex consiste nell'utilizzare la proprietà `FlashVars`, che passa brevi stringhe di testo a un documento Flex 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 Flex 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 Flex Remoting sono i più versatili per il caricamento di dati esterni, ma presentano la curva di apprendimento più impegnativa. Per informazioni su Flex Remoting, visitare il sito www.macromedia.com/support/flexremoting.

`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\Flex 8\Samples and Tutorials\Samples\ActionScript\LoadText.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 come 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 deselegionare il set di caratteri facendo clic su di esso.
6. Per eliminare tutti i set di caratteri selezioni 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.

ATTENZIONE

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 **charembd 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 "helo 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 in 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>\First Run\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 `glyphRange` 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 adipiscing
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\Flex 8\Samples and Tutorials\Samples\ActionScript\FlexType.

In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 8/Samples and Tutorials/Samples\ActionScript/FlexType.

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 **antialiastype 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\Flex 8\Samples and Tutorials\Samples\ActionScript\FlexType.

In Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 8/Samples and Tutorials/Samples/ActionScript/FlexType.

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.

NOTA

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 **advancedaatable 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 definisce 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 = 0xFFFFFF;

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 imposta 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 = ""
```

NOTA

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 *selettore*, a una o più proprietà di stile e ai rispettivi valori. Il seguente stile definisce ad esempio un selettore 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\Flex 8\Samples and Tutorials\Samples\ActionScript\LoadText.

Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 ommesso e il carattere che segue è maiuscolo.

Proprietà CSS	Proprietà ActionScript	Uso e valori supportati
<code>text-align</code>	<code>textAlign</code>	I valori riconosciuti sono <code>left</code> , <code>center</code> , <code>right</code> e <code>justify</code> .
<code>font-size</code>	<code>fontSize</code>	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
<code>text-decoration</code>	<code>textDecoration</code>	I valori riconosciuti sono <code>none</code> e <code>underline</code> .
<code>margin-left</code>	<code>marginLeft</code>	Viene utilizzata solo la parte numerica del valore. Le unità (px, pt) non vengono analizzate; i pixel e i punti sono equivalenti.
<code>margin-right</code>	<code>marginRight</code>	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\Flesh 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");
```

NOTA

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 caricano 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\Flex 8\Samples and Tutorials\Samples\ActionScript\LoadText.

Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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;
```

NOTA

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
p	Influisce su tutti i tag <code><p></code> .
body	Influisce su tutti i tag <code><body></code> . Lo stile p, se specificato, ha la priorità rispetto allo stile body .
li	Influisce su tutti i tag di punto elenco <code></code> .
a	Influisce su tutti i tag di ancoraggio <code><a></code> .
a:link	Influisce su tutti i tag di ancoraggio <code><a></code> . Questo stile viene applicato dopo ogni stile a.
a:hover	Applicato a un tag di ancoraggio <code><a></code> quando il mouse viene spostato sul collegamento. Questo stile viene applicato dopo ogni stile a e a:link. Quando il mouse viene spostato fuori dal collegamento, lo stile a:hover viene eliminato dal collegamento.
a:active	Applicato a un tag <code><a></code> di ancoraggio facendo clic con il mouse sopra il collegamento. Questo stile viene applicato dopo ogni stile a e a:link. Quando viene rilasciato il pulsante del mouse, lo stile a:active 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><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>
```

NOTA

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.

7. 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.

8. Salvare il file di testo come **story.xml**.
9. 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 ``, 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 = "<font color='#FF0000'>This is red text</font>";
```
- **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 = "<font face='Times, Times New Roman'>Displays as  
either Times or Times New Roman...</font>";
```

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 = "<font size='24' color='#0000FF'>This is blue, 24-  
point text</font>";
```

Aniché 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, creata 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> 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)
>	> (maggiore di)
&	& (e commerciale)
"	" (virgolette doppie)
'	' (apostrofo, virgolette singole)

Flash supporta inoltre codici di caratteri espliciti, 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.

NOTA

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.

NOTA

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 Scorrevole 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 > Scorrevole.
- 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 > Scorrevole.

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.

NOTA

Se il testo *non* viene caricato all'interno del file SWF in modo dinamico, selezionare Testo > Scorrevole 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Strings.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 `Locate` 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.msp
- <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.

NOTA

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 "virgoletta inglese" o il carattere "virgoletta 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
<code>\b</code>	Il carattere barra rovesciata.
<code>\f</code>	Il carattere di avanzamento pagina.
<code>\n</code>	Il carattere di avanzamento riga.
<code>\r</code>	Il carattere di ritorno a capo.
<code>\t</code>	Il carattere Tab.
<code>\unnnn</code>	Il carattere Unicode con il codice di carattere specificato mediante il numero esadecimale <i>nnnn</i> . Il codice <code>\u263a</code> , ad esempio, corrisponde al carattere smile.
<code>\xnn</code>	Il carattere ASCII con il codice di carattere specificato mediante il numero esadecimale <i>nn</i> .
<code>\'</code>	Una virgoletta semplice.
<code>\"</code>	Una virgoletta doppia.
<code>\\</code>	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
```

AVVISO

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 imporre il confronto rigorosi 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Strings.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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 virgole 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Strings.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 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\Flex 8\Samples and Tutorials\Samples\ActionScript\Strings.
- Su Macintosh, accedere a *Macintosh HD*/Applicazioni/Macromedia Flex 8/Samples and Tutorials/Samples\ActionScript/Strings.

```

        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 numOfDay:Number = dayArr.length;
for (var i = 0; i<numOfDay; 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.Utls {  
    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.Utills;
var randomMonth:Number = Utills.randomRange(0, 11);
var min:Number = Utills.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utills.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 compila 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 virgole in un elenco di argomenti.

L'utilizzo di spazi dopo le virgole rende più semplice distinguere tra le chiamate dei metodi e le parole chiave, come illustrato nell'esempio seguente:

```
function addItem(item1:Number, item2:Number):Number {
    return (item1 + item2);
}
var sum:Number = addItem(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.

NOTA

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 && veggio == leek) {}
```

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

```
if ((fruit == apple) && (veggio == 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` predefinito 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:

Numero errore	Testo del messaggio
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.

Numero errore	Testo del messaggio
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.

Numero errore	Testo del messaggio
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.

Numero errore	Testo del messaggio
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	Circolarità 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> .

Numero errore	Testo del messaggio
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 <code>AS</code> 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 <code>compile time</code> .
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 <code>for-in</code> è <code>String</code> .
1200	Una funzione setter non può restituire un valore.

Numero errore	Testo del messaggio
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.

Operatore	Descrizione	Associatività
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() + "\tACSII: " + 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

Tasto della lettera o del numero	Codice tasto	Codice tasto ASCII
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`:

Tasto del tastierino numerico	Codice tasto	Codice tasto ASCII
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

Tasto del tastierino numerico	Codice tasto	Codice tasto ASCII
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`:

Tasto funzione	Codice tasto	Codice tasto ASCII
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 <code>ActionScript</code> .	Questo tasto è riservato dal sistema e non può essere utilizzato in <code>ActionScript</code> .
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

D

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.

NOTA

Per informazioni sulla versione di Flash Player più utilizzata nei diversi Paesi, vedere 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

E

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:

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

NOTA

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.
```

Anziché 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

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 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

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 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.

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.

Identificatori: 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 String. 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 *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()`.

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 ActionScript 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
 - referimenti 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
- contenitore 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
 - tracce 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
- uplicazione, 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 anche video

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
- inidificati 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 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
 - incapsulamento
 - 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 HTTP, 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
MediaPlayer, 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
messa in 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_DoFSCCommand, 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 incapsulamento 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. *Vedere* modifica in scala a 9 porzioni
- Scheda Controllo, Debugger 777
- Scheda Proprietà, Debugger 779
- scheda Variabili, Debugger 776
- scorrevolesse
 - 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
- Specifiche 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
 - imposizione 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
 - specifica delle impostazioni per 48
 - uso 48
 - visualizzazione manuale 51
- suoni
 - Vedere anche* contenuti multimediali esterni
- super (prefisso) 818
- superclasse 281
- superfici
 - 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 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

