


Caricare e scaricare un form dalla memoria potrebbe sembrare un'operazione molto semplice ma nasconde più di un'insidia. È fondamentale ricordarsi che esistono sei operazioni fondamentali legate all'apertura e chiusura di un form; le sei operazioni sono:

1. Inizializzazione
Corrisponde all'operazione di generazione, ovvero l'[istanza](#), di una nuova copia della [classe](#) base da cui il form nasce. È identificata dall'evento  **Initialize** e costituisce parte dell'operazione di [allocazione](#).
2. Caricamento
Corrisponde all'effettivo caricamento di tutti i dati in memoria, in particolare dell'interfaccia grafica del form. È identificato dall'[evento](#) **Load**.
3. Visualizzazione
Avviene nel momento in cui l'interfaccia grafica è mostrata sullo schermo ed in situazioni normali segue automaticamente l'operazione di caricamento.
4. Celazione
Non è regolata da alcun evento. È tuttavia possibile da codice nascondere un form senza tuttavia per questo scaricarlo dalla memoria. Nei forms [modali](#) determina il ritorno alla procedura che ha richiesto la visualizzazione modale.
5. Scaricamento
Corrisponde alla chiusura del form tramite casella di controllo, pulsante di chiusura oppure mediante istruzione Unload. È identificato dagli eventi **QueryUnload** e **Unload**.
6. Terminazione
L'ultima operazione prima dell'effettiva distruzione del form in memoria. È invocata quando tutti i riferimenti ad un'istanza sono [deallocati](#).

Tenendo a mente queste sei operazioni possiamo vedere che esistono tre differenti modalità di istanza di un form:

1. Riferimento alla classe base
È possibile fare riferimento al nome della classe del form per istanziare automaticamente una copia del form con il nome della classe che l'ha generato. Ad esempio è possibile fare riferimento alla classe Form2 trattandola come una normale istanza di tale classe.
2. Dichiarazione di un [oggetto](#) e successiva istanza esplicita

Dichiarando una variabile oggetto della classe del form è possibile quindi creare (istanziare) più copie dello stesso form e scaricarle dalla memoria quando non servono più.

Differentemente a ciò che si potrebbe pensare i forms non si distruggono automaticamente quando la variabile oggetto termina la sua vita uscendo dalla porzione di codice in cui essa è dichiarata. La ragione è data dalla **duplice vita dei forms**, argomento affrontato più avanti.

3. Dichiarazione di un oggetto ed istanza in maniera implicita

È possibile istanziare automaticamente una copia della classe del form nella stessa dichiarazione. In tal modo l'oggetto non sarà allocato nel momento in cui viene dichiarato ma solo quando viene richiesto un [metodo](#) o un [membro](#) dell'oggetto.

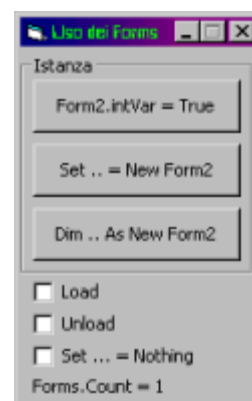
Queste due differenti argomentazioni danno vita a tantissime combinazioni di funzionamento e malfunzionamento.

A tal scopo è stato sviluppato un progetto dimostrativo che a scelta dell'utente carica un form secondario e opzionalmente lo carica, lo scarica e lo dealloca dalla memoria.


L'interfaccia si presenta molto semplice ed intuitiva: sono presenti tre pulsanti di nome **cmdCaso** (con indici da 0 a 3) in grado di generare un riferimento al form da utilizzare nelle tre differenti maniere viste in precedenza.

Al di sotto dei pulsanti sono presenti 3 caselle di controllo per la scelta di tre operazioni possibili: caricamento, scaricamento e deallocazione.

In fondo al form è presente invece una semplice etichetta che mostrerà sullo schermo il numero di forms caricati in memoria. Il numero più corretto, se tutto funziona a dovere è 1 e corrisponde al form che stiamo utilizzando. Un valore di 2 o superiore indicherà che uno o più form non sono stati scaricati dalla memoria e continuano a vivere vita propria, pur se al di fuori delle nostre possibilità.



Il form che verrà caricato è straordinariamente semplice: consta infatti di un unico pulsante in grado di nascondere il form modale aperto. Questo comporterà quindi il ritorno alla procedura che ne ha richiesto l'apertura.

Nello stesso form è presente il pulsante di chiusura  di Windows per permettere anche lo scaricamento del form piuttosto che la semplice celazione.

Il secondo form, di nome **Form2**, contiene al suo interno una variabile pubblica di nome **intVar** che servirà per analizzare il comportamento del form in funzione delle operazioni richieste e verrà inizializzata a 0 nell'evento *Initialize* del form.

I quattro eventi *Initialize*, *Load*, *Unload* e *Terminate* del form sono regolate dall'apparizione di una finestra di messaggio informativa sul valore della variabile **intVar** oppure sul numero di forms caricati in memoria.

Ritorniamo al primo form: la sua vita si compone dell'unico evento Click per tutti e tre i pulsanti, differenziati dal valore del loro Indice. La routine di gestione dell'evento verificherà quale pulsante è stato premuto ed in base a questo aprirà il form nella maniera richiesta: tramite riferimento alla classe base, con dichiarazione di un oggetto e successiva istanza oppure con istanza automatica.

Per tutti e tre i casi, comunque saranno eseguite due operazioni: incremento della variabile membro `intVar` e visualizzazione in maniera modale del secondo form.


In aggiunta a questo l'utente potrà scegliere se eseguire altre tre operazioni: caricamento del form prima dell'incremento della variabile, scaricamento del form dopo la celazione e deallocazione dell'istanza del form al termine dell'uso.

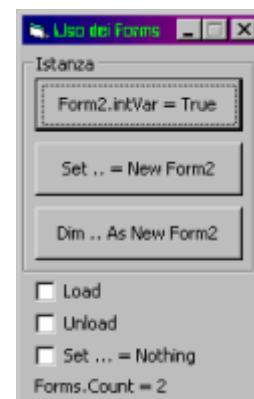
Si raccomanda un test di sperimentazione con tutte le soluzioni possibili, caricando il form più di una volta e vedere quindi qual'è la procedura più corretta per il risultato che si vuole ottenere.

In questa sede verranno presentato solo un paio di casi, i più lampanti e strani.

Il primo di questi utilizza il primo dei tre pulsanti senza selezionare alcuna delle caselle di controllo in basso.

Alla prima pressione del pulsante possiamo notare che sono eseguire nell'ordine le seguenti operazioni:

1. Evento Initialize ed `intVar` vale 0
2. Incremento della variabile `intVar` a 1
3. Evento Load ed `intVar` continua a valere 1
4. Visualizzazione del secondo form e seguono due scelte:
 - a. Celazione del form mediante il pulsante Nascondi
 - b. Chiusura del form mediante il pulsante di chiusura Comporta lo scaricamento del form dalla memoria
5. Ritorno al form principale



In questo primo caso possiamo analizzare due comportamenti strani: il primo riguarda la celazione del form mediante il pulsante nascondi. Il form non viene scaricato dalla memoria ed il numero di forms risulta essere 2.

Premendo nuovamente il primo pulsante il form viene nuovamente mostrato, il valore di **intVar** viene incrementato ma non scatta alcun evento poiché il form era ancora in memoria ed il nuovo valore di **intVar** non è mostrato.

Qualcuno dirà: "fin qui nulla di strano", infatti il form era in memoria, non è stato scaricato e quindi è giusto che non venga né ricaricato né scaricato; il valore di **intVar** può tranquillamente incrementare a 2 o numeri successivi.

Riapriamo nuovamente il secondo form premendo il primo dei tre pulsanti e stavolta, invece di nascondendolo semplicemente chiudiamolo mediante il pulsante di chiusura sulla barra del titolo.

Qui iniziano le faccende preoccupanti: il form è scaricato regolarmente, scatta l'evento Unload e l'etichetta sul primo form torna a segnare il numero 1; il secondo form infatti non è più in memoria. Un'ulteriore riapertura del secondo form ci mostrerà quindi qualcosa di

sconvolgente: l'evento *Initialize* non scatta più, ma scatta soltanto l'evento *Load*. E non finisce qui: il secondo form, pur essendo stato scaricato dalla memoria continua a mantenere il suo valore nella variabile **intVar**, che sembra essere diventata di tipo *Static*, cioè in grado di mantenere il suo valore anche tra differenti chiamate.

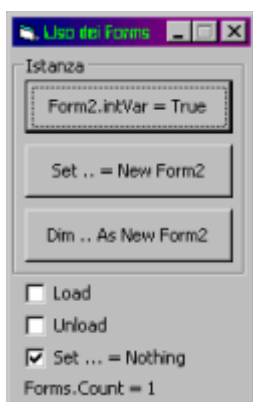
Esiste tuttavia una spiegazione logica al problema ed è legata alla doppia vita dei forms accennata in precedenza. Abbiamo infatti visto che il riferimento al secondo form è istanziato (evento *Initialize*) in occasione della prima apertura dello stesso e seguono quindi normali caricamenti (evento *Load*) e scaricamenti (evento *Unload*). Il nodo mancante è quindi la distruzione (evento *Terminate*) dell'istanza creata automaticamente.

La doppia vita dei form è quindi determinata da *Initialize/Terminate* e da *Load/Unload*. Nonostante sembri normale pensare che lo scaricamento del form dalla memoria ne rimuova tutti i riferimenti non è così. Tale doppia vita è dimostrata anche dal fatto che, se viene incrementato il valore di una variabile quale **intVar**, il form non viene caricato in memoria, ma soltanto istanziato se non è già avvenuto. Possiamo forzare il caricamento con l'istruzione **Load** oppure facendo uso di una proprietà o di un metodo intrinseco del form. Il richiamo di un metodo non nativo di un form non forza comunque il caricamento dello stesso, a meno che il metodo non faccia uso di elementi nativi del form. Si può quindi sintetizzare che tutti gli elementi nativi del form fanno parte della vita regolata da *Load/Unload*, mentre tutti i membri aggiuntivi fanno parte della prima vita del form, regolata da *Initialize/Terminate*.

Per scaricare l'istanza automatica è necessario chiudere il form che l'ha generata ovvero il primo form.

Altra cosa buffa: l'evento *Terminate* dell'istanza automatica scatta solo dopo lo stesso evento del primo form ovvero è deallocato prima il primo form e poi il secondo, nonostante quest'ultimo sia stato generato dal primo dei due form.

Il secondo caso strano che analizzeremo riguarda anch'esso la doppia vita dei forms ed è in un certo senso l'opposto del caso visto in precedenza.



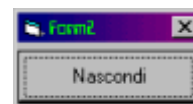
Dopo aver selezionato la casella di controllo **Set ... = Nothing**, che si occupa di deallocare l'istanza del form in memoria, utilizziamo nuovamente il primo pulsante per richiedere l'istanza automatica del secondo form.

In maniera analoga alla precedente possiamo notare che scatta il primo evento *Initialize* e la variabile **intVar** vale 0; in seguito scatta l'evento *Load* e la variabile **intVar** vale 1. Il form è quindi mostrato sullo schermo.

Se chiudiamo tale form mediante il pulsante di chiusura sulla barra del titolo possiamo osservare lo scaricamento del form (evento *Unload*), seguito dalla deallocazione dell'istanza automatica. Il numero di form in memoria è quindi l'iniziale 0. Il programma è stato ripristinato allo stato iniziale; se rilanciamo l'apertura del secondo form in maniera analoga gli eventi sono scatenati nella stessa sequenza della precedenza.

Proviamo invece a non chiudere il secondo form dal suo pulsante di chiusura ma nascondiamolo semplicemente con il pulsante Nascondi.

Il numero di forms in memoria è 2, come nel caso precedente; il form infatti non è stato scaricato e non è scattato l'evento Unload. Un'ulteriore pressione del primo pulsante farà sussultare più di un cuore...

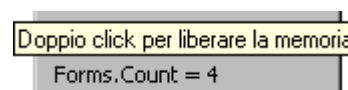


Scatta infatti nuovamente l'evento Initialize nonostante il secondo form sia ancora in memoria e non siano scattati in precedenza né l'evento Unload né quello Terminate. Ed ancora il valore della variabile `intVar` è ripristinato a zero dall'evento Initialize. Proviamo a nascondere anche questo secondo form e notiamo che il contatore dei forms caricati in memoria continua ad aumentare; adesso segna il numero 3.

Questa è una bruttissima situazione di instabilità: è caricato in memoria almeno un form secondario ma non è possibile avere un riferimento diretto. L'uso dell'istanza automatica con il nome della classe genera infatti un'ulteriore form in memoria.

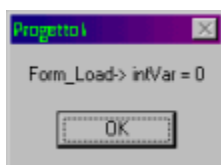
Anche per questa stranezza esiste una spiegazione. Selezionando la casella **Set ... = Nothing** abbiamo fatto sì che venga deallocata la variabile oggetto ad istanza automatica, ovvero abbiamo distrutto la prima delle due vite del form. Si attende la fine della seconda vita, quella regolata da Load/Unload per distruggere completamente il form e lanciare l'ultimo evento (Terminate) della prima vita.

È possibile ristabilizzare la situazione chiudendo il programma oppure facendo doppio click sull'etichetta in fondo al primo form. Funziona anche con il caso precedente ma non era stato accennato per sottolineare l'aspetto dello scaricamento delle istanze disordinato.



L'ultimo caso non è per nulla strano ma lo si vuole sottolineare per evitare possibili errori durante lo sviluppo.

Dopo aver selezionato la casella di controllo **Load**, premere un pulsante qualsiasi tra i tre disponibili.



Saranno generati in sequenza gli eventi Initialize e Load ma stavolta, a differenza delle situazioni precedenti il valore di **intVar** sarà 0.

La spiegazione è molto semplice. L'istruzione Load forza il caricamento del form cui segue quindi l'evento Load che nel nostro caso mostrerà il valore della variabile **intVar**, prima ancora che le venga assegnato il nuovo valore.



Si raccomanda un'analisi del codice per chiarire le sequenze di esecuzioni delle istruzioni e comprendere come evitare situazioni poco piacevoli ristabilizzando quindi l'armonia delle due vite separate degli oggetti Form.



[Torna all'indice delle Stranezze](#)
