# Creazione di un controllo Winsock in un modulo di classe

http://www.vbsimple.net/cliserv/clser_07.htm

**Difficoltà: 3 / 5**

Il problema in esame trova molteplici soluzioni. La più semplice, anche se non molto pulita consiste nel creare un form, inserire all'interno d'esso il controllo Winsock ed impostare la proprietà **Visible** del form a False. In tal modo il form sarà caricato in memoria ma sarà completamente invisibile all'utente. Per utilizzare il controllo basterà fare riferimento al form contenitore.
Ad esempio se il nostro form nascosto si chiama Form1 ed il controllo Winsock si chiama Winsock1, basterà scrivere Form1.Winsock1 per ottenere un riferimento completo al controllo.

Per sfruttare in maniera diretta il controllo all'interno del form si potrebbe eseguire un assegnamento di una variabile di classe al controllo all'interno del form.
In codice tutto questo si effettua con queste semplici istruzioni:

```
 1. Public WithEvents Winsock1 As Winsock
 2. Private FormWSock As Form1
 3.
 4. Private Sub Class_Initialize()
 5.     Set FormWSock = New Form1
 6.     Set Winsock1 = FormWSock.Winsock1
 7. End Sub
 8.
 9. Private Sub Class_Terminate()
10.     Set Winsock1 = Nothing
11.     Unload FormWSock
12.     Set FormWSock = Nothing
13.     MsgBox "Classe1 deallocata"
14. End Sub
```

La prima riga definisce la variabile membro collegata al controllo del Form1.
La seconda riga prepara una variabile utilizzabile per creare istanze del Form1. Avremmo potuto usare la normale istruzione Load Form1, ma non avremmo potuto utilizzare più di un'istanza di classe per volta.

All'interno dell'inizializzazione della classe avviene l'istanza del Form1 nella variabile FormWSock e subito dopo il collegamento (mapping) tra la variabile Winsock1 della classe con il controllo Winsock1 del form istanziato da FormWSock (righe 5 e 6).

Alla chiusura della classe avviene la deallocazione della variabile membro Winsock1, lo scaricamento del form dalla memoria, la deallocazione della variabile puntatore al Form1, ed un messaggio di confermata chiusura, che può essere tranquillamente rimosso.

Questa classe è facilmente istanziabile e possiede una variabile pubblica che contiene il nostro controllo Winsock.

 

Una seconda soluzione al problema è data dall'istruzione *CreateObject*, la quale crea un'istanza di un controllo ActiveX e restituisce un puntatore ad un Object.
Il suo sfruttamento presuppone la conoscenza del nome interno della classe. Nel caso del Winsock il suo nome è **MSWinsock.Winsock**. Tutti i nomi delle classi sono memorizzate nel registro di Windows ed ogni classe possiede un proprio CLSID.

Questa classe sarà molto più semplice della precedente ma possiede numerose limitazioni:

```
 1. Public Winsock1 As Object
 2.
 3. Private Sub Class_Initialize()
 4.     Set Winsock1 = CreateObject("MSWinsock.Winsock")
 5. End Sub
 6.
 7. Private Sub Class_Terminate()
 8.     Set Winsock1 = Nothing
 9.     MsgBox "Classe2 deallocata"
10. End Sub
```

Innanzitutto possiamo subito notare alla prima riga che il controllo Winsock1 è dichiarato del tipo generico Object, perché l'istruzione CreateObject (riga 4) restituisce un puntatore ad una variabile Object. È assolutamente inefficace ogni tentativo di assegnare tale puntatore ad una variabile di tipo Winsock.

Lo svantaggio principale di questa soluzione sta nella mancanza di supporto nella scrittura del codice. Infatti se la variabile fosse stata dichiarata di tipo Winsock l'IDE di Visual Basic avrebbe in automatico suggerito i nomi e la sintassi dei metodi e delle proprietà. Purtroppo la tipologia Object non fornisce questo supporto.
Altro svantaggio consiste nel non poter utilizzare gli eventi legati al controllo. Infatti la tipologia Object non permette l'utilizzo della parola chiave WithEvents che fornisce l'accesso agli eventi di un controllo.

Una caratteristica molto particolare di questa soluzione appare nella mancanza di riferimenti nel file di progetto. Infatti, a differenza di tutte le altre soluzioni, non sarà obbligatorio aprire la finestra di dialogo Componenti o quella Riferimenti dal menu progetto per inserire un collegamento alla libreria MSWINSCK.OCX.

---

Una terza soluzione consiste nell'inserire un riferimento al file MSWINSCK.OCX tramite la voce Riferimenti del menu Progetto. La differenza principale rispetto ai normali riferimenti è che non utilizzeremo il file presentatoci da Visual Basic, ovvero MSWINSCK.OCA, ma il suo genitore, MSWINSCK.OCX. Naturalmente non troveremo questo file nella lista di riferimenti di Visual Basic; basterà premere il tasto sfoglia, cercarsi tale file e premere OK.

Una volta inserito tale riferimento potremo scrivere all'interno del modulo di classe:

```
Public WithEvents Winsock1 As New Winsock
```

In tal modo la variabile **Winsock1** conterrà un riferimento completo ad un oggetto Winsock già istanziato. Potranno pure essere utilizzati gli eventi associati a questo controllo.

 

L'unico svantaggio presente in questa soluzione è che non potranno essere inseriti controlli di tipo Winsock sopra i forms del progetto. Infatti, nel momento in cui tenteremo di aggiungere il componente Winsock dal menu Progetto, ci apparirà *Nome già in uso*. Il motivo è semplice: mediante il riferimento precedente, Visual Basic ha già caricato la classe MsWinsock.Winsock. Il tentativo di inserire sia il riferimento, sia il componente Winsock genera una classe con lo stesso nome di quella chiamata per prima, generando un errore di nome non univoco. Perché ricordiamo che Visual Basic nell'utilizzo dei componenti esterni inserisce un riferimento al file con estensione OCA.

In ogni caso si consiglia l'utilizzo della prima soluzione in problemi di questo genere, tenendo conto che, fintanto che sarà necessario utilizzare il controllo Winsock, dovrà restare in memoria il form contenitore del controllo.
Se il programma usufruitore della classe scarica accidentalmente il form dalla memoria il codice non funzionerà nel modo corretto.

Questo problema non sorge invece con la seconda soluzione poiché non vi sono form caricati in memoria, ovvero potenziali rischi, ma il tutto risiede in poche righe di classe.

La terza soluzione è sicuramente la più comoda, ma limita l'utilizzo della classe all'interno di altri progetti. Il che va contro il principio della riutilizzabilità del codice, testata d'angolo della programmazione ad oggetti.

Fibia FBI
17 Novembre 2000

Torna all'indice Client / Server

 