Aggiungere un menu al menu di sistema

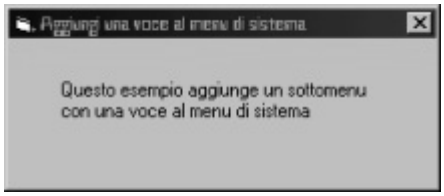
http://www.vbsimple.net/howto/ht_027.htm

Difficoltà: 4 / 5

In un HowTo abbiamo visto come richiamare il menu di sistema di un form ed in un altro abbiamo visto come disabilitare mediante la cancellazione una voce del menu di sistema.
L'ultima operazione da effettuare, quindi, è l'aggiunta di una voce o di un sottomenu al menu di sistema. Per effettuare questa operazione utilizzeremo un paio di funzioni API.

In questo progetto avremo bisogno di un form e di un modulo standard in cui inseriremo la *Window Procedure* che riceverà i messaggi inviati al form. Infatti per poter controllare il click sulla voce di menu aggiunta dovremo subclassare il form creando una funzione di gestione dei messaggi.

Cominciamo a vedere il form. Al suo interno non c'è nessun controllo. Noi abbiamo inserito soltanto una *Label* esplicativa, ma essa non è necessaria in alcun modo.



Il codice del form è il seguente:

```
 1. Option Explicit
 2.
 3. Private Declare Function CreatePopupMenu Lib "USER32" () As Long
 4. Private Declare Function DestroyMenu Lib "USER32" (ByVal hMenu As Long) As Long
 5. Private Declare Function AppendMenu Lib "USER32" Alias "AppendMenuA" (ByVal hMenu
    As Long, ByVal wFlags As Long, ByVal wIDNewItem As Long, ByVal lpNewItem As String)
    As Long
 6. Private Declare Function GetSystemMenu Lib "USER32" (ByVal hWnd As Long, ByVal
    bRevert As Long) As Long
 7. Private Declare Function SetWindowLong Lib "USER32" Alias "SetWindowLongA" (ByVal
    hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
 8. Private Const MF_POPUP = &H10&
 9. Private Const MF_SEPARATOR = &H800&
10. Private Const MF_STRING = &H0&
11. Private Const GWL_WNDPROC = -4
12. Private newMenu As Long
13.
```

Alla riga 3 abbiamo dichiarato la funzione *CretePopupMenu* che crea ed alloca un nuovo menu di popup e ne restituisce l'handle. La riga successiva effettua l'operazione inversa, deallocando e distruggendo un menu.

Alla riga 5 abbiamo la dichiarazione della funzione *AppendMenu*. Essa servirà per aggiungere un menu ad un menu esistente. Tuttavia, essa si occuperà soltanto di collegare i due menu, non gestendoli in alcun modo.

Alla riga 6 abbiamo la funzione *GetSystemMenu*, vista in un altro HowTo, che restituisce

l'handle del menu di sistema. Ad esso infatti, si dovrà aggiungere il nuovo menu.

L'ultima funzione API è la *SetWindowLong*, una funzione molto complessa e pericolosa (riga 7). Essa effettua varie operazioni sulle finestre e nel nostro caso provvederà a collegare la nuova *Window Procedure* per effettuare il subclassing del form.

Seguono (righe 8-10) tre costanti (**MF_POPUP**, **MF_SEPARATOR** e **MF_STRING**) che indicano il tipo di operazione da effettuare con AppendMenu.

La costante **GWL_WNDPROC** alla riga 11 comunicherà alla funzione *SetWindowLong* di collegare una nuova Window Procedure.

Abbiamo anche una variabile

```
14. Private Sub Form_Load()
15.     Dim SysMenu As Long
16.     SysMenu = GetSystemMenu(Me.hWnd, 0&)
17.     newMenu = CreatePopupMenu
18.     AppendMenu newMenu, MF_STRING, IDM_INFOS, "Informazioni"
19.     AppendMenu SysMenu, MF_SEPARATOR, 0&, vbNullString
20.     AppendMenu SysMenu, MF_POPUP, newMenu, "Informazioni"
21.     lProcOld = SetWindowLong(Me.hWnd, GWL_WNDPROC, AddressOf MenuHandler)
22. End Sub
23.
```

Al caricamento del form avviene la creazione del nuovo menu e il subclassing del form. Alla riga 16 ritroviamo l'handle del menu di sistema del form e lo memorizzeremo nella variabile **SysMenu**.

Alla riga 17 verrà creato un nuovo menu a comparsa. Il suo handle sarà memorizzato nella variabile **newMenu** (essa è dichiarata all'interno del modulo standard). Nella riga successiva creeremo una nuova voce di menu nel nuovo menu. Utilizzeremo la funzione AppendMenu, passandole come parametri l'handle del menu in cui verrà inserita la voce, il tipo di menu (***MF_STRING***), l'identificatore che verrà restituito come messaggio alla Window Procedure (***IDM_INFOS***) ed il testo *"Informazioni"* che apparirà all'utente.

Questo nuovo menu andrà inserito nel menu di sistema. Così, alla riga 19, aggiungeremo prima una voce di separazione (***MF_SEPARATOR***) al menu di sistema **SysMenu**. Subito sotto questa incolleremo il nuovo menu impostando come testo d'esso *"Informazioni"*.

In pratica abbiamo creato un sottomenu il cui handle è newMenu ed impostato per esso la Caption *"Informazioni"*. All'interno di questo menu abbiamo inserito una voce di menu con il testo *"Informazioni"*. Nel momento in cui l'utente clicca su questa voce di menu viene lanciato il messaggio che indica che è stato premuta una voce di menu e verrà specificato come *"responsabile"* della chiamata la costante **IDM_INFOS**.

Per rintracciare però questo messaggio, sarà necessario subclassare il form specificando l'indirizzo di una funzione che fungerà da Window Procedure. Poiché l'operatore *AddressOf* richiede il nome di una funzione a livello di modulo, abbiamo dovuto inserire la nostra funzione in un modulo standard.

Così, alla riga 21, viene effettuato il subclassing: la funzione *SetWindowLong* con il parametro ***GWL_WNDPROC*** specifica che desideriamo creare una nuova Window

Procedure. L'indirizzo della nuova Window Procedure è indicato da *AddressOf* **MenuHandler**.

Così facendo, tutti i messaggi inviati dal form, passeranno **prima** dalla funzione **MenuHandler**, che si occuperà di verificarli e reagire di conseguenza.

L'indirizzo della Window Procedure originale - che deve essere ripristinato in chiusura del programma - viene salvato nella variabile pubblica **lProcOld**.

```
24. Private Sub Form_Unload(Cancel As Integer)
25.     SetWindowLong Me.hWnd, GWL_WNDPROC, lProcOld
26.     DestroyMenu newMenu
27. End Sub
```

Così, nello scaricamento del form, ripristiniamo la Window Procedure originale, passandole come indirizzo il puntatore salvato precedentemente.
Una volta scollegato il menu, possiamo provvedere alla sua distruzione e deallocazione dalla memoria, mediante la funzione *DestroyMenu*.

Il nostro form termina qui. Possiamo vedere l'unica funzione di cui si compone il modulo, che funzionerà da Window Procedure.

```
1. Option Explicit
2.
3. Private Declare Function CallWindowProc Lib "USER32" Alias "CallWindowProcA" (ByVal
   lpPrevWndFunc As Long, ByVal hWnd As Long, ByVal Msg As Long, ByVal wParam As Long,
   ByVal lParam As Long) As Long
4. Public Const WM_SYSCOMMAND = &H112
5. Public Const IDM_INFOS As Long = 1001
6. Public lProcOld As Long
7.
```

Poiché la nostra Window Procedure riceverà tutti i messaggi inviati dal form, è necessario richiamare la Window Procedure originale, per effettuare tutte le altre operazioni (il movimento del mouse, il ridisegno, la pressione sopra il form, gli spostamenti, etc...).

A tal scopo abbiamo dichiarato la funzione API *CallWindowProc* che richiama una Window Procedure di cui conosciamo l'indirizzo (riga 3).

Alla riga 4 abbiamo la costante costante ***WM_SYSCOMMAND*** che identifica i messaggi inviati dal menu di sistema. Segue la costante IDM_INFOS, inventata, che indica il numero identificativo della voce di menu creata nel form.

Alla riga 6 risiede la dichiarazione della variabile lProcOld che dovrà contenere il puntatore alla Window Procedure originale, da richiamare per tutti i messaggi e per ripristinare il programma in uscita.

```
 8. Public Function MenuHandler(ByVal hWnd As Long, ByVal iMsg As Long, ByVal wParam As
    Long, ByVal lParam As Long) As Long
 9.     If iMsg = WM_SYSCOMMAND Then
10.         If wParam = IDM_INFOS Then
11.             MsgBox "Questa voce è stata aggiunta al menu di sistema",
    vbInformation, "Informazioni"
12.             Exit Function
13.         End If
14.     End If
15.     MenuHandler = CallWindowProc(lProcOld, hWnd, iMsg, wParam, lParam)
```



```
16. End Function
```

La funzione **MenuHandler** è la nostra Window Procedure. La definizioe dei parametri è obbligatoria in questa forma per tutte le Window Procedure.

Se il messaggio inviato proviene dal menu di sistema (costante ***WM_SYSCOMMAND***), sarà necessario verificare se è stata scelta la voce ***IDM_INFOS***, ovvero il nostro menu. In tal caso verrà mostrata una finestra informativa con il testo ***"Questa voce è stata aggiunta al menu di sistema"***. Fatto questo la Window Procedure ha terminato la sua funzione.

Se invece la voce di menu non è quella da noi creata, oppure il messaggio non riguarda il menu di sistema, sarà necessario chiamare la Window Procedure originale, per permettere il corretto funzionamento del form.

La chiamata alla funzione originale avviene tramite la funzione CallWindowProc e specificando come indirizzo il puntatore **lProcOld**, salvato al momento del subclassing.

Il nostro codice termina qui. Possiamo provare il nostro programma e vederne gli effetti.

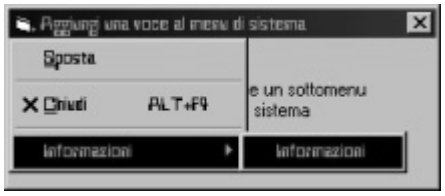
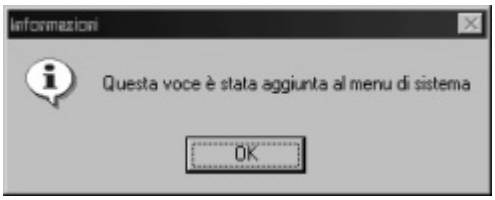


Figura 2

Nel momento in cui apriremo il menu di sistema troveremo un sottomenu e dentro d'esso una voce di menu. Il click sopra d'essa mostrerà una finestra informativa.



Il progetto contiene molti rischi e molte trappole.
In ogni caso non interrompere mai il programma tramite la funzione interna all'IDE di Visual Basic oppure tramite la funzione End, perché esse non si preoccupano di rimettere in ordine le Window Procedure.

Se le chiamate sono gestite correttamente il progetto funzione, ma un semplice errore può trasformarsi in un blocco del computer.

Fibia FBI
15 Febbraio 2001

Torna all'indice degli HowTo

