Corso Intermedio - Lezione 11

http://www.vbsimple.fbi/intermed/int_11.htm

Vedremo in questa lezione come aggiungere alcuni menu a discesa. Sulla barra degli strumenti standard è presente l'icona per il richiamo dell'Editor dei menu (la stessa opzione è richiamabile attraverso il menu *Strumenti*) che è già stato presentato nella Lezione 16 del Corso Base.

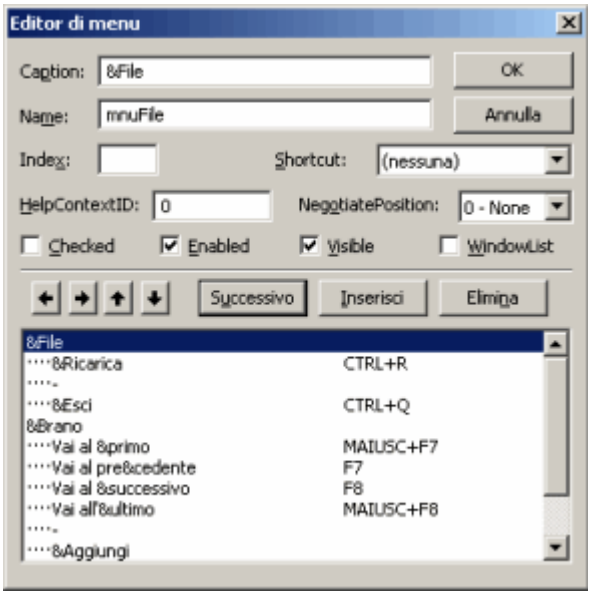


Figura 1

Aggiungeremo al nostro semplice programma alcuni menu per consentire un'utilizzo più semplice ed aggiungere nuove caratteristiche non raggiungibili mediante i pulsanti sopra il form; a tal fine avviamo l'Editor dei menu ed inseriamo le seguenti voci di menu:

&File	mnuFile	
&Ricarica	mnuFile_Ricarica	CTRL+R
-	mnuFile_Separatore	
&Esci	mnuFile_Esci	CTRL+Q
Bra&no	mnuBrano	
Vai al &primo	mnuBrano_Sposta(0)	MAIUSC+F7
Vai al pre&cedente	mnuBrano_Sposta(1)	F7

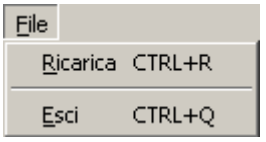


Figura 2

Vai al &successivo	mnuBrano_Sposta(2)	F8
Vai all'&ultimo	mnuBrano_Sposta(3)	MAIUSC+F8
-	mnuBrano_Separatore	
&Aggiungi	mnuBrano_Cmd(0)	
&Modifica	mnuBrano_Cmd(1)	
&Elimina	mnuBrano_Cmd(2)	

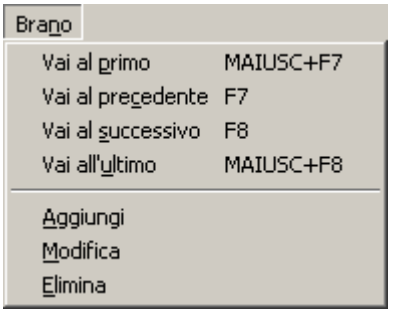


Figura 3

Il carattere **&** nelle proprietà *Caption* identifica il tasto utilizzato come combinazione rapida di tasti. Ad esempio il menu con l'etichetta &File consente all'utente di premere i tasti ALT+F e richiamare in automatico la voce del menu. Un'etichetta contenente soltanto il carattere - (trattino) come nel menu *mnuFile_Separatore* identifica un separatore, cioè una voce fissa di menu che non può essere selezionata.

Al momento ci limiteremo ad attivare le voci già implementate, cioè quelle del menu ***Brano***, semplicemente richiamando le routine che gestiscono l'evento corrispondente, come segue:

```
 1. Private Sub mnuBrano_Cmd_Click(Index As Integer)
 2.     Select Case Index
 3.         Case 0: cmdAggiungiBrano.Value = True
 4.         Case 1: cmdModificaBrano.Value = True
 5.         Case 2: cmdEliminaBrano.Value = True
 6.     End Select
 7. End Sub
 8.
```

Come spiegato in un HowTo l'assegnazione del valore ***True*** alla proprietà **Value** di un CommandButton genera l'evento *Click* e così del codice assegnato. In tal modo potremo gestire con estrema semplicità le azioni Aggiungi, Modifica ed Elimina di un brano.

```
 9. Private Sub mnuBrano_Sposta_Click(Index As Integer)
10.     With lstBrani
11.         Select Case Index
12.             Case 0: .ListIndex = 0
13.             Case 1: .ListIndex = .ListIndex - 1
14.             Case 2: .ListIndex = .ListIndex + 1
15.             Case 3: .ListIndex = .ListCount - 1
16.         End Select
17.     End With
18. End Sub
19.
```

La scelta del brano è fatta semplicemente modificando l'elemento selezionato nell'elenco. Il primo brano corrisponde all'indice 0, mentre l'ultimo brano corrisponde al numero di elementi -1. La modifica della proprietà **ListIndex** provoca anche l'esecuzione dell'evento *Click*.

Questo codice tuttavia non è sicuro: non è effettuato alcun controllo sul valore della proprietà ListIndex prima di incrementarla o decrementarla ed in caso di supero del valore minimo o massimo si genererà un errore. Per tale ragione è opportuno sincronizzare lo

stato dei pulsanti e delle voci di menu con la situazione corrente.

Non si tratta di un'operazione complessa ma piuttosto noiosa; ogni volta che un pulsante deve essere disabilitato è necessario disabilitare anche la corrispondente voce del menu. Per questa ragione sarà utilizzata una nuova tecnica per semplificare il problema: le proprietà personalizzate del form. Sarà infatti aggiunta una nuova proprietà al form per abilitare o disabilitare il controllo corrispondente.

Non volendo scrivere tre diverse proprietà con contenuti molto simili, si è preferito scrivere un'unica proprietà cui fornire un valore in entrata. In funzione del valore in entrata (corrispondente allo stato del pulsante da gestire) sarà assegnato un valore al pulsante corrispondente. Volendo semplificare ulteriormente il problema si è preferito aggiungere un'enumerazione  e far riferimento a tali valori anziché a degli incomprensibili Indice 0, 1 o qualcos'altro.

Dichiariamo pertanto in cima al codice del form, nella sezione dichiarazioni la seguente enumerazione:

```
 1. Private Enum enumComandoPulsanti
 2.     ComandoAggiungiBrano = 0
 3.     ComandoModificaBrano = 1
 4.     ComandoEliminaBrano = 2
 5. End Enum
 6.
```

Tali valori saranno passati alla proprietà StatoComandi dedicata allo scopo:

```
 7. Private Property Let StatoComandi(ByVal Comando As enumComandoPulsanti, ByVal Stato
    As Boolean)
 8.     Select Case Comando
 9.         Case ComandoAggiungiBrano: cmdAggiungiBrano.Enabled = Stato
10.         Case ComandoModificaBrano: cmdModificaBrano.Enabled = Stato
11.         Case ComandoEliminaBrano: cmdEliminaBrano.Enabled = Stato
12.     End Select
13.     mnuBrano_Cmd(Comando).Enabled = Stato
14. End Property
15.
```


L'operazione effettuata alla riga 13 funziona perché a differenza di altri linguaggi le enumerazioni sono anche semplici valori numerici che non richiedono conversioni di tipo e gli indici del nostro menu corrispondono ai valori dell'enumerazione.

Basterà assegnare un valore booleano a questa proprietà per aggiornare contemporaneamente la proprietà Enabled del menu e del pulsante corrispondente, quindi sostituendo l'assegnazione alla proprietà Enabled dei singoli pulsanti come segue:

```
16. Private Sub cmdAggiungiBrano_Click()
17.     ... Codice trattato nelle lezioni precedenti ...
18.     txtAnnoPubblicazione.Locked = Not blnModifica
19.     txtNote.Locked = Not blnModifica
20.     StatoComandi(ComandoEliminaBrano) = Not blnModifica
21.     StatoComandi(ComandoModificaBrano) = Not blnModifica
22. End Sub
23.
24. Private Sub cmdEliminaBrano_Click()
25.     ... Codice trattato nelle lezioni precedenti ...
26.     lstBrani.RemoveItem lstBrani.ListIndex
27.     StatoComandi(ComandoEliminaBrano) = False
```



```
28.     StatoComandi(ComandoModificaBrano) = False
29.     ... Codice trattato nelle lezioni precedenti ...
30. End Sub
31.
32. Private Sub cmdModificaBrano_Click()
33.     ... Codice trattato nelle lezioni precedenti ...
34.     txtNote.Locked = Not blnModifica
35.     StatoComandi(ComandoEliminaBrano) = Not blnModifica
36.     StatoComandi(ComandoAggiungiBrano) = Not blnModifica
37. End Sub
38.
39. Private Sub Form_Load()
40.     ... Codice trattato nelle lezioni precedenti ...
41.     txtAnnoPubblicazione.MaxLength = 4
42.     StatoComandi(ComandoModificaBrano) = False
43.     StatoComandi(ComandoEliminaBrano) = False
44.     blnModifica = False
45.     ... Codice trattato nelle lezioni precedenti ...
46. End Sub
47.
```

Esistono numerose maniere per gestire situazioni come quella esposta e l'uso delle proprietà rappresenta la più semplice e caratteristica. Per sincronizzare invece lo stato delle voci di menu utilizzate per spostarsi al brano scelto l'uso di una proprietà non è la maniera migliore. Si preferirà usare una subroutine dedicata ad aggiornare lo stato di tutti e quattro le voci di menu:

```
48. Private Sub AggiornaMenuSposta()
49.     With lstBrani
50.         mnuBrano_Sposta(0).Enabled = .ListIndex > 0
51.         mnuBrano_Sposta(1).Enabled = mnuBrano_Sposta(0).Enabled
52.         mnuBrano_Sposta(2).Enabled = .ListIndex < .ListCount - 1
53.         mnuBrano_Sposta(3).Enabled = mnuBrano_Sposta(2).Enabled
54.     End With
55. End Sub
56.
```

La routine **AggiornaMenuSposta** aggiorna lo stato di tutte e quattro le voci del menu Brano e pertanto basterà inserire il suo richiamo all'interno della routine delegata alla gestione dell'evento Click dell'elenco lstBrani:

```
57. Private Sub lstBrani_Click()
58.     ... Codice trattato nelle lezioni precedenti ...
59.         End With
60.         StatoComandi(ComandoEliminaBrano) = True
61.         StatoComandi(ComandoModificaBrano) = True
62.     End If
63.     If lstBrani.Enabled Then AggiornaMenuSposta
64. End Sub
65.
```

La stessa routine AggiornaMenuSposta sarà aggiunta subito dopo la fase di inserimento e di cancellazione di un brano in modo da avere situazioni coerenti.

Sorge però una piccola complicazione ma abbastanza semplice da aggirare: nel momento in cui l'utente richiede l'inserimento di un nuovo brano o la modifica di uno preesistente sarà necessario bloccare anche l'opzione di scelta di un brano dal menu e anziché duplicare il codice di disattivazione utilizzeremo un'altra proprietà del form:

```
66. Private Property Let StatoMenuSposta(ByVal Stato As Boolean)
67.     lstBrani.Enabled = Stato
68.     mnuBrano_Sposta(0).Enabled = Stato
```



```
69.      mnuBrano_Sposta(1).Enabled = Stato
70.      mnuBrano_Sposta(2).Enabled = Stato
71.      mnuBrano_Sposta(3).Enabled = Stato
72.      If Stato Then AggiornaMenuSposta
73.  End Property
74.
```

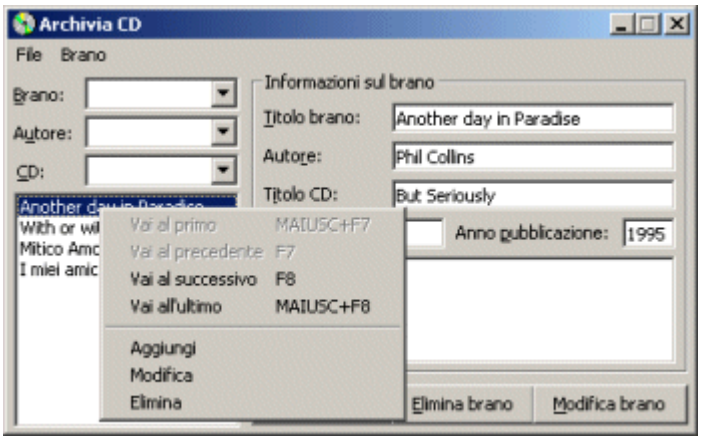
Basterà quindi sostituire tutte e due le occorrenze di `lstBrani.Enabled = blnModifica` con `StatoMenuSposta = Not blnModifica`.

L'uso delle scorciatoie nelle voci di menu consente un accesso più rapido alle funzionalità più utilizzate. Ad esempio per passare dal brano corrente al precedente basterà premere il tasto F7, mentre per passare al successivo basterà F8.

Per rendere ancor più moderno il comportamento della nostra applicazione potremmo richiamare il menu **Brano** al click con il tasto destro sopra la ListBox. Questo genere di menu a comparsa su richiesta dell'utente è detto Popup e può essere richiamato con semplicità:

```
75.  Private Sub lstBrani_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
     As Single)
76.      If Button = vbLeftButton Then
77.          lstBrani_Click
78.      ElseIf Button = vbRightButton Then
79.          PopupMenu mnuBrano
80.      End If
81.  End Sub
```

La funzione PopupMenu richiede la specifica del menu da far apparire; esso conterrà naturalmente tutte le voci interne. Utilizzata così la funzione mostra il menu come rappresentato nella figura a fianco.



Tuttavia la funzione PopupMenu consente di specificare anche le coordinate in cui il menu dovrà apparire e se una voce di menu dovrà apparire in grassetto. Ad esempio il codice

```
PopupMenu mnuBrano, , , , mnuBrano_Sposta(2)
```

mostra il menu popup alle coordinate del puntatore del mouse ed evidenzia in grassetto la voce ***Vai al successivo***.

Tale impostazione non ha alcun valore particolare a livello di programma ma risalta maggiormente all'occhio all'utente e solitamente indica l'azione predefinita con un eventuale doppio click sul controllo, ma il suo funzionamento deve essere previsto a programma.

Vedremo nella prossima lezione l'uso del menu File ed introdurremo le prime funzioni di ricerca e filtro.

Fibia FBI
15 Febbraio 2004

Torna alla decima lezione Vai alla dodicesima lezione
