



[Home Page](#)

[Informazioni](#)

[Aiuto](#)

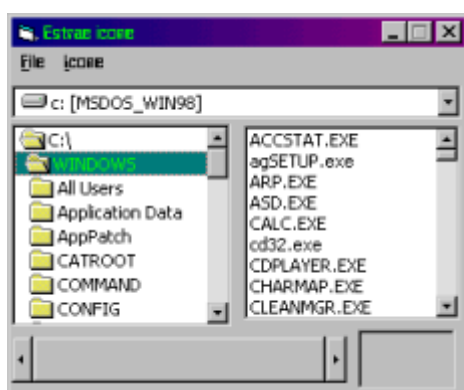
Estrarre le icone da un file

http://www.vbsimple.net/howto/ht_049.htm

Difficoltà: 2 / 5

Abbiamo visto in [un altro progetto](#) come creare un [file di risorse](#) contenente testi ed immagini da compilare assieme al resto di un programma. Vedremo in questo articolo invece come estrarre le icone da un file già compilato, ad esempio un eseguibile oppure una [DLL](#).

Per eseguire quest'operazione svilupperemo un semplicissimo progetto con un unico form che utilizza tre funzioni [API](#) complementari: la prima è *ExtractIconEx* che si occuperà di estrarre una o più icone da un file; la seconda è *DrawIconEx* che consente di disegnare - riprodurre - un'icona sopra un [Device Context](#); l'ultima funzione è *DestroyIcon* che [deallocherà](#) la memoria utilizzata dall'icona estratta e da rappresentare. Passiamo direttamente allo sviluppo del form che costituisce l'interfaccia utente del nostro progetto:



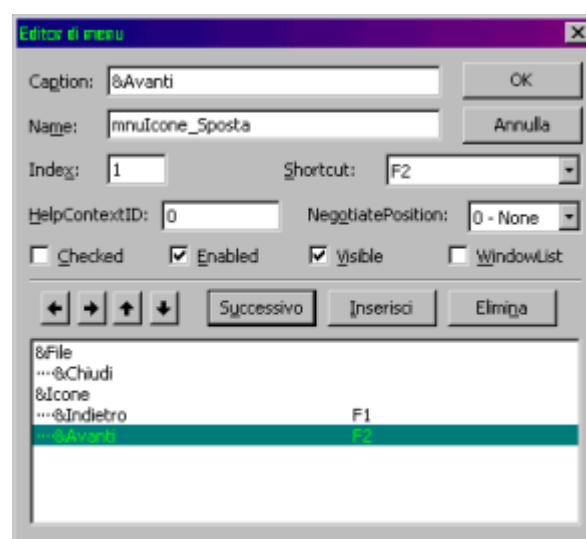
Il form si compone di cinque semplici controlli più due menu, non obbligatori ma utilizzati per pura comodità.

La parte superiore della superficie del form è occupata da tre controlli: un *DriveListBox* di nome **drvDrive**, un *DirListBox* di nome **dirCartelle** ed un *FileListBox* di nome **filFiles**. Naturalmente i tre controlli sono sincronizzati per indicare rispettivamente un'unità disco, la struttura di cartelle ed i corrispondenti files contenuti nella cartella attuale.

Nella parte inferiore del form sono presenti una barra di scorrimento orizzontale di nome **hsbIcona** ed una *PictureBox* di nome **picIcone**.

La barra dei menu invece conterrà due semplici menu costruiti con l'Editor di Menu posto sulla barra degli strumenti standard.

Il menu **File** di nome **mnuFile** conterrà al suo interno una voce di nome **mnuFile_Chiudi** con il testo **Chiudi** (anzi **&Chiudi**). Il menu **Icone** invece prende il nome di **mnuIcone** e contiene due voci di menu: **Indietro** ed **Avanti**, entrambi associati al nome **mnuIcone_Sposta** ma differenziati per il loro indice. Ad entrambe le voci è associato un tasti scorciatoia: F1 per Indietro ed F2 per Avanti.



I due menu in realtà non sono obbligatori nello sviluppo della nostra applicazione ma sono utilizzati per semplificare la visualizzazione delle icone estratte consentendo l'uso di tasti scorciatoia.

Il codice si presenta altrettanto semplice:

```

1. Option Explicit
2.
3. Private Const DI_MASK = &H1
4. Private Const DI_IMAGE = &H2
5. Private Const DI_NORMAL = DI_MASK Or DI_IMAGE
6.
7. Private Declare Function ExtractIconEx Lib "SHELL32" Alias "ExtractIconExA" (ByVal
    lpzFile As String, ByVal nIndex As Long, phiconLarge As Long, phiconSmall As
    Long, ByVal nIcons As Long) As Long
8. Private Declare Function DrawIconEx Lib "USER32" (ByVal hdc As Long, ByVal xLeft As
    Long, ByVal yTop As Long, ByVal hIcon As Long, ByVal cxWidth As Long, ByVal cyWidth
    As Long, ByVal istepIfAniCur As Long, ByVal hbrFlickerFreeDraw As Long, ByVal
    diFlags As Long) As Long
9. Private Declare Function DestroyIcon Lib "USER32" (ByVal hIcon As Long) As Long
10.
```

La [costante](#) `DI_NORMAL` alla riga 3 si compone della somma delle altre due costanti API dichiarate in precedenza. La prima di queste costanti (`DI_MASK`) indica, nell'operazione di disegno dell'icona, di riprodurre la sola maschera che produrrà quindi la trasparenza dell'icona stessa. La seconda costante (`DI_IMAGE`) indica di riprodurre l'immagine vera e propria. Pertanto la somma delle due costanti (`DI_NORMAL`) indica la riproduzione dell'icona comprensiva di immagine e maschera di trasparenza.

Alla riga 7 è dichiarata la funzione API *ExtractIconEx* che, partendo dal nome del file specificato nell'argomento *lpzFile* restituisce il numero di icone contenute nello stesso oppure restituisce una coppia di icone (grande e piccola) contenute all'interno del file. Il differente comportamento è regolato dall'argomento *nIconIndex*: nel caso esso sia -1, la funzione restituirà il numero di icone presenti nel file; per tutti gli altri valori maggiori o uguali a 0 sarà restituita l'icona corrispondente all'indice scelto.

La funzione *DrawIconEx* dichiarata alla riga 8 disegna l'icona specificata nell'argomento *hIcon* sul [Device Context](#) indicato da *hdc*, alle coordinate e grandezze indicate. Utilizza le costanti API viste in precedenza per determinare se disegnare la sola immagine, la sola maschera oppure la combinazione di entrambe.

L'ultima funzione API dichiarata alla riga 9 è *DestroyIcon* che consente di deallocare la memoria dall'icona caricata e liberare l'[handle](#) ad essa assegnato.

Definiamo una nuova funzione che si rivela molto utile in svariate situazioni in cui si devono maneggiare percorsi di files o cartelle. Il suo nome è **LastBSlash**, ad indicare Last Back Slask ovvero Ultima barra rovesciata.

```

11. Private Function LastBSlash(ByVal Path As String, ByVal Slash As Boolean) As String
12.     If Slash And (Right$(Path, 1) <> "\") Then Path = Path & "\"
13.     If Not Slash And (Right$(Path, 1) = "\") Then Path = Left$(Path, Len(Path) - 1)
14.     LastBSlash = Path
15. End Function
16.
```

La funzione consente di aggiungere o rimuovere l'ultima barra rovesciata (\) al percorso

Path indicato, in base al valore dell'argomento **Slash**. Se viene richiesta l'aggiunta della barra rovesciata (**Slash** impostato su True) verrà verificato che quel percorso non termini già con una barra del genere ed in tal caso verrà aggiunta con una semplice concatenazione mostrata alla riga 12.

Viceversa se viene richiesta la rimozione della barra rovesciata verrà verificata prima la presenza di quest'ultima e nel caso essa esista verrà rimossa alla riga 13.

La funzione risulta utile quando si maneggiano nomi di cartelle per le quali è necessario avere o meno la barra rovesciata finale. L'esempio più comune è quello di concatenare il nome di una cartella al nome di un file per ottenere quindi il percorso completo del suddetto file. In tal situazione il nome della cartella deve necessariamente terminare con una barra rovesciata. Potrebbe sembrare scontato che basti concatenare una barra rovesciata ed il nome del file al nome della cartella ma non è sempre così. Difatti se il nome della cartella termina già con una barra rovesciata si verificherà un errore in quanto il nome della cartella conterrà due barre rovesciate. Ad esempio:

```
Cartella = "C:\MiaCartella"
File = "documento.txt"
Percorso = Cartella & "\" & File
```

Produrrà il risultato voluto, ovvero **Percorso** = "C:\MiaCartella\documento.txt"...
...ma se..

```
Cartella = "C:\"
File = "documento.txt"
Percorso = Cartella & "\" & File
```

Produrrà invece un percorso errato, corrispondente a "C:\\documento.txt"
La funzione LastBSlash consentirà invece di evitare errori di questo genere.

```
Percorso = LastBSlash(Cartella, True) & File
```

La barra rovesciata al termine della stringa **Cartella** sarà comunque presente e potrà essere concatenata al nome del file che segue.

Seguono quindi le tre funzioni che regolano l'[evento](#) ⚡ Click sopra i controlli di tipo *DriveListBox*, *DirListBox* e *FileListBox*. I tre controlli infatti dovranno essere sincronizzati tra loro poiché se l'utente decide di utilizzare un'unità disco differente, la struttura delle cartelle dovrà rispecchiare la nuova unità disco selezionata e lo stesso dovrà fare l'elenco con i files.

```
17. Private Sub drvDrive_Change()
18.     Dim strOldPath As String
19.     strOldPath = filFiles.Path
20.     On Error Resume Next
21.     dirCartelle.Path = Left$(drvDrive.Drive, 2)
22.     If Err.Number <> 0 Then
23.         Err.Clear
24.         drvDrive.Drive = strOldPath
25.         dirCartelle.Path = strOldPath
26.     End If
27. End Sub
28.
```

La prima routine gestirà il comportamento dell'evento Click sul controllo **drvDrive** per la

scelta di un'unità disco. In questa routine è fondamentale non sottovalutare il caso che l'utente scelga, anche involontariamente, un'unità disco non pronta ovvero che non possa essere letta perché non disponibile o più semplicemente perché non è presente un disco in tale unità. Un errore comunissimo è quello di lasciare che l'utente scelga l'unità a dischetti A: ma non vi sia alcun disco all'interno del drive da leggere. Se tale situazione non viene gestita correttamente il programma termina con un errore.

Alla riga 18 è dichiarata la variabile **strOldPath** che andrà a contenere il percorso attuale indicato dalla proprietà *Path* dell'elenco dei files (riga 19). Infatti in caso di un errore, quale un disco non pronto, verrà ripristinato il percorso precedente.

Alla riga 20 è aggiunta un'istruzione di gestione di errori; il verificarsi di uno di questi comporterà il salto all'istruzione successiva. Con questa minima gestione possiamo quindi modificare il percorso del controllo **dirCartelle** per uniformarlo all'unità disco scelta nel controllo **drvDrive**. Non sarà necessario modificare il percorso del controllo **filFiles** poiché di questo si occuperà la corrispondente routine di gestione dell'evento Click del controllo **dirCartelle**.

Pertanto, sia che si verifichi un errore nella scelta dell'unità disco, sia che non si verifichi, il flusso del programma arriverà alla riga 22 nella quale esiste un controllo del codice di errore generato. Nel caso non fosse stato generato alcun errore, il codice corrisponderà a 0 e le righe 23-25 non verranno eseguite. In caso contrario sarà innanzitutto cancellato quell'errore (riga 23) per non propagarlo ad altre routine e poi verranno ripristinati i percorsi precedenti dei controlli **drvDrive** e **dirCartelle**, salvati nella variabile **strOldPath**.

```
29. Private Sub dirCartelle_Change()  
30.     filFiles.Path = dirCartelle.Path  
31. End Sub  
32.
```

La sincronizzazione dei controlli **dirCartelle** e **filFiles** si presenta molto più semplice. A differenza del controllo **drvDrive**, non sarà possibile scegliere un percorso inaccessibile (quei salvo rari casi di malfunzionamento dell'unità disco o sistemi di protezione contro l'accesso non autorizzato, ma non ci occuperemo di questi casi).

Ci basterà pertanto impostare il percorso del controllo **filFiles** al percorso del controllo **dirCartelle**, in seguito al quale sarà aggiornato automaticamente l'elenco dei files. Questa procedura scatta automaticamente anche quando è modificata l'unità disco, poiché abbiamo visto che la routine precedente a questa modifica anche il percorso del controllo **dirCartelle**, in seguito alla quale (riga 25) scatterà l'aggiornamento del controllo **filFiles**.

```
33. Private Sub filFiles_Click()  
34.     hsbIcona.Enabled = False  
35.     hsbIcona.Value = 0  
36.     hsbIcona.Max = ExtractIconEx(LeftBSlash(filFiles.Path, True) &  
    filFiles.filename, -1, ByVal 0&, ByVal 0&, ByVal 0&)  
37.     If hsbIcona.Max > 0 Then  
38.         hsbIcona.Max = hsbIcona.Max - 1  
39.         hsbIcona.Enabled = True  
40.     End If  
41.     hsbIcona_Change  
42. End Sub  
43.
```

Nel momento in cui l'utente seleziona un elemento dal controllo **filFiles** il file dovrà essere analizzato per verificare se contiene al suo interno una o più icone al suo interno.

L'operazione è eseguita alla riga 36 utilizzando la funzione *ExtractIconEx* fornendo ad essa il nome del file da cui estrarre le icone e specificando il valore -1 come argomento **nIconIndex**. La funzione restituirà il numero di icone presenti nel file che sarà impostato nella proprietà **Max** del controllo **hsbIcona**, disabilitato alla riga 34.

Se è presente almeno un'icona nel file, il numero di icone sarà decrementato di 1 per riadattarlo alla base 0 del controllo **hsbIcona** e seguirà la riabilitazione alla riga 39.

Alla riga 41 è richiamata la funzione *hsbIcona_Change*, corrispondente all'evento *Change* del controllo *HScrollBar* che vedremo subito dopo. Questo servirà per mostrare la prima icona estratta dal file.

```

44. Private Sub hsbIcona_Change()
45.     Dim hLargeIcon As Long
46.     Dim hSmallIcon As Long
47.     Set picIcone.Picture = Nothing
48.     If ExtractIconEx>LastBSlash(filFiles.Path, True) & filFiles.filename,
        hsbIcona.Value, hLargeIcon, hSmallIcon, 1) > 0 Then
49.         Call DrawIconEx(picIcone.hdc, 0, 0, hLargeIcon, ByVal 0&, ByVal 0&, ByVal
        0&, ByVal 0&, DI_NORMAL)
50.         Call DrawIconEx(picIcone.hdc, 35, 0, hSmallIcon, ByVal 0&, ByVal 0&, ByVal
        0&, ByVal 0&, DI_NORMAL)
51.         DestroyIcon hSmallIcon
52.         DestroyIcon hLargeIcon
53.     End If
54. End Sub
55.

```

Nel momento in cui il valore della barra di scorrimento cambia perché l'utente sceglie una nuova icona oppure perché viene recuperata la prima icona dal file (vedi riga 41), l'icona corrispondente al valore del controllo **hsbIcona** dovrà essere estratta dal file e visualizzata nel controllo **picIcone**.

Alle righe 45 e 46 sono dichiarate due variabili di tipo Long per contenere l'[handle](#) delle due icone da estrarre (quella grande e quella piccola).

La prima operazione da eseguire è la cancellazione dell'eventuale immagine precedente nella casella **picIcone** (riga 47) e quindi potranno essere estrarre le due icone dal file specificando l'indice dell'icona da estrarre, corrispondente al valore della barra di scorrimento (riga 46). Le icone estratte saranno mantenute in memoria e sono riportati gli handle alla rispettive nei valori **hLargeIcon** e **hSmallIcon**. Se le icone sono estratte con successo la funzione *ExtractIconEx* restituisce valore maggiore di zero.

Sarà quindi necessario visualizzare le due icone estratte e l'operazione è effettuata mediante la funzione *DrawIconEx* che consente di incollare un'icona o una parte di essa sopra un [Device Context](#). L'operazione è effettuata alle righe 49 e 50 sopra il DC della casella **picIcone**. La prima icona sarà incollata alle coordinate 0,0 mentre la seconda leggermente più avanti, alle coordinate 35,0. L'operazione, abbiamo già detto, utilizzerà la costante **DI_NORMAL** che comprende l'immagine dell'icona e la sua maschera di trasparenza.

Non appena riprodotte le due icone sarà nostro compito distruggerle e deallocare la

memoria ad esse riservata e l'operazione è eseguita mediante la funzione DestroyIcon alle righe 51 e 52.

```

56. Private Sub mnuFile_Chiudi_Click()
57.     Unload Me
58. End Sub
59.
60. Private Sub mnuIcone_Sposta_Click(Index As Integer)
61.     Dim intIcona As Integer
62.     intIcona = hsbIcona.Value
63.     If Index = 0 Then intIcona = intIcona - 1 Else intIcona = intIcona + 1
64.     If intIcona < 0 Then intIcona = 0
65.     If intIcona > hsbIcona.Max Then intIcona = hsbIcona.Max
66.     If intIcona = hsbIcona.Value Then hsbIcona_Change Else hsbIcona.Value =
        intIcona
67. End Sub

```

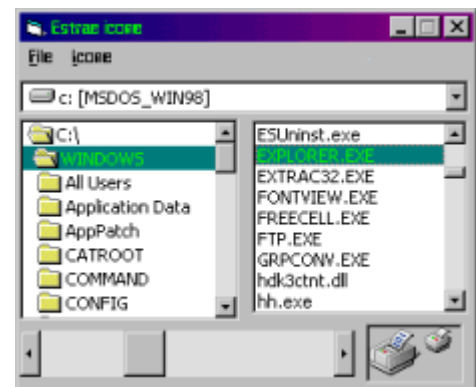
Le ultime due routine serviranno per gestire il comportamento dei due menu: il primo è mnuFile_Chiudi e si occuperà soltanto di chiudere l'applicazione (riga 57); il secondo di questi menu è mnuIcone_Sposta e consente di scegliere l'icona precedente o quella successiva al valore specificato dalla barra di scorrimento.

Si occuperà di modificare il valore della barra di scorrimento in avanti o indietro a seconda dell'indice passato all'evento. Index = 0 riguarda l'icona precedente mentre Index = 1 riguarda l'icona successiva. Saranno effettuati una serie di controlli per mantenere l'icona entro il limite massimo di icone presenti nel file e sarà quindi mostrata l'icona corrispondente (riga 66).

La dimostrazione del progetto è davvero molto semplice: basterà selezionare un file dal controllo FileListBox per vedere apparire la prima delle icone presenti nel file.

Utilizzando la barra di scorrimento in basso sarà possibile scorrere le singole icone nel file.

Per ogni icona sarà mostrata l'immagine grande e quella piccola. Utilizzando i tasti funzione F1 e F2 (purché siano stati assegnati in fase di progettazione nell'editor di Menu) sarà possibile scorrere le icone nel file con estrema semplicità.



Il progetto si presenta abbastanza semplice e scorrevole e non presenta difetti o rischi di funzionamento insiti nel codice. L'unica possibile limitazione riguarda la struttura di Windows 9x; in particolare si potrebbe assistere allo sgradevole effetto di esaurimento delle risorse grafiche dovuto all'eccessivo uso di icone.

Fibia FBI
18 Agosto 2002

 [Torna all'indice degli HowTo](#)