



## Creare un file di collegamento

[http://www.vbsimple.net/activity/act\\_19.htm](http://www.vbsimple.net/activity/act_19.htm)

Richiesta di: [Alessandro](#) - 14 Febbraio 2002

Difficoltà: ►►► 3 / 5

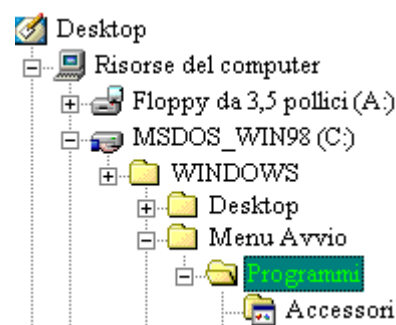
*Come posso creare un file di collegamento in Visual Basic?*

*Dovrei realizzare lo stesso comando **Crea collegamento** che si fa con il tasto destro del mouse quando si trascina un file da una cartella ad un'altra o al Desktop.*

Esistono due soluzioni per effettuare questa operazione: una molto semplice ma altrettanto limitata ed una molto più complessa e completa. Il codice della seconda soluzione è presente nel CD di Visual Basic alla cartella **ShellLnk** ma non è affatto commentato; in quest'articolo tratteremo invece la soluzione più semplice che utilizzerà la [libreria](#) utilizzata dal Kit di installazione dei progetti, talvolta chiamato **Autocomposizione Installazione** (per Visual Basic 5) e talvolta **Creazione guidata pacchetti di installazione** o **Package and Deployment Wizard** (per Visual Basic 6).

La limitazione principale riguarda l'impossibilità di scegliere un'icona differente da quella a cui il collegamento punta. Esiste anche un'altra apparente limitazione che obbliga il programmatore a fare riferimento alla cartella in cui è contenuta la cartella Programmi del Menu Avvio del computer. È possibile superare quest'ultima limitazione in una maniera che vedremo in seguito.

Ciò che bisogna tenere in mente è che il collegamento sarà fatto in maniera relativa alla cartella Programmi del Menu Avvio. In una struttura come quella mostrata nella figura a fianco, se dovessimo creare un collegamento nella cartella Programmi del Menu Avvio, il percorso da utilizzare sarà un semplice "." in quanto cartella di riferimento.



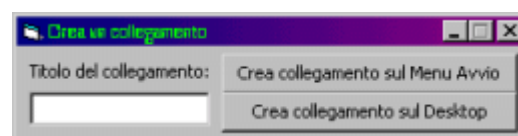
Se dovessimo invece creare il collegamento sul Desktop sarà necessario utilizzare come percorso relativo "..\..\Desktop".

Il livello superiore ("..\") indica appunto il Menu Avvio, al di fuori della cartella Programmi e l'ulteriore livello superiore ("..\..\") indicherà la cartella Windows.

Il progetto che svilupperemo in quest'articolo si compone di un solo form con 4 semplici controlli: una *Label* a puro scopo descrittivo di nome

**lblTitoloCollegamento**, una *TextBox* **txtTitoloCollegamento** di nome

**txtTitoloCollegamento** in cui l'utente potrà inserire il titolo ovvero la caption del collegamento creato. A fianco di questi avremo due semplici *CommandButton* di nome **cmdCollegamentoMenuAvvio** e **cmdCollegamentoDesktop** che effettueranno il



collegamento sulle due cartelle indicate.

Sono presentate tre soluzioni, una per ogni versione di Visual Basic, dalla 4 alla 6 ognuna delle quali utilizza la sua libreria di riferimento. Poiché le tre funzioni [API](#) hanno lo stesso nome avremmo preferito utilizzare le istruzioni di compilazione condizionale ma purtroppo il [prototipo](#) della funzione (leggi la dichiarazione) è leggermente diverso; per tale ragione utilizzeremo una funzione che in base alla versione di Visual Basic richiama richiama la funzione relativa.

```
1. Option Explicit
2.
3. Private Declare Function VB4CreateShellLink Lib "STKIT432.DLL" Alias
  "fCreateShellLink" (ByVal lpstrFolderName As String, ByVal lpstrLinkName As String,
  ByVal lpstrLinkPath As String, ByVal lpstrLinkArgs As String) As Long
4. Private Declare Function VB5CreateShellLink Lib "VB5STKIT.DLL" Alias
  "fCreateShellLink" (ByVal lpstrFolderName As String, ByVal lpstrLinkName As String,
  ByVal lpstrLinkPath As String, ByVal lpstrLinkArgs As String) As Long
5. Private Declare Function VB6CreateShellLink Lib "VB6STKIT.DLL" Alias
  "fCreateShellLink" (ByVal lpstrFolderName As String, ByVal lpstrLinkName As String,
  ByVal lpstrLinkPath As String, ByVal lpstrLinkArgs As String, ByVal fPrivate As
  Long, ByVal sParent As String) As Long
```

Alle righe 3-5 sono presentate le 3 funzioni [API](#) *fCreateShellLink* che si differenziano per il nome della libreria utilizzata: **STKIT432.DLL** per la versione 4 a 32 bit di VB, **VB5STKIT.DLL** per la versione 5 di VB ed infine **VB6STKIT.DLL** per la versione 6 di VB. In più l'ultima versione richiede due parametri aggiuntivi non presenti nelle altre due versioni. Per tale ragione abbiamo ridefinito i nomi delle funzioni API tramite degli [Alias](#).

Le funzioni che prima si chiamavano *fCreateShellLink* all'interno del nostro progetto hanno i nomi *VB4CreateShellLink*, *VB5CreateShellLink* e *VB6CreateShellLink*.

Le funzioni API *fCreateShellLink* nelle versioni 4 e 5 richiedono il passaggio di 4 parametri stringa: il primo è il percorso della cartella in cui creare il collegamento, **relativa** alla cartella Programmi del Menu Avvio; il secondo è il titolo del collegamento ovvero il nome che avrà il file di collegamento; il terzo parametro indica il percorso completo del file da collegare e l'ultimo parametro infine consente il passaggio di ulteriori argomenti al programma da richiamare ([vedi a tal scopo l'HowTo dedicato alla lettura dei parametri sulla riga di comando](#)). L'ultima versione utilizza gli stessi parametri ma ne richiede due ulteriori di nessuna importanza. 😊

```
6. Private Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal
  lpLibFileName As String) As Long
7. Private Declare Function FreeLibrary Lib "kernel32" (ByVal hLibModule As Long) As
  Long
8.
9. Private Enum VersioneVB
10.     VisualBasicUnknown = 0
11.     VisualBasic4 = 4
12.     VisualBasic5 = 5
13.     VisualBasic6 = 6
14. End Enum
15.
16. Private AppPath As String
17.
```


Alla riga 6 abbiamo dichiarato la funzione API *LoadLibrary* che utilizzeremo per determinare automaticamente quale versione delle funzioni API *fCreateShellLink*

utilizzare. La funzione *LoadLibrary* infatti consente di caricare in memoria una [DLL](#) specificandone il nome. La funzione *FreeLibrary* alla riga 7 consente l'operazione inversa ovvero lo scaricamento dalla memoria di una DLL caricata.

Alle righe 9-14 si presenta una nuova [enumerazione](#): **VersioneVB** che utilizzeremo per specificare la versione di VB in uso per l'utilizzo della funzione API *fCreateShellLink* relativa oppure lasciare fare alla nostra funzione di ricerca in caso di valore **VisualBasicUnknown**.

Alla riga 16 abbiamo dichiarato una variabile di nome **AppPath** che conterrà il percorso del file eseguibile del nostro progetto.

```
18. Private Function CreateShellLink(ByVal intVersione As VersioneVB, ByVal strGruppo
    As String, ByVal strTitolo As String, ByVal strPath As String) As Boolean
19.     Dim intConta As Integer
20.     Dim lngRis As Long
21.
22.     On Error GoTo Err_Handler
23.     If intVersione = VisualBasicUnknown Then
```

La nostra funzione *CreateShellLink* si occuperà di richiamare la funzione API *fCreateShellLink* richiesta oppure determinerà in maniera automatica quale utilizzare. Essa richiede il passaggio di un valore dell'[enumerazione](#)  **VersioneVB** ed i tre parametri minimi da fornire alla funzione API.

Alla riga 22 abbiamo inserito un semplice gestore degli errori per evitare il blocco del programma nel caso che la DLL specificata non esista. Se invece il programma si affida alla ricerca automatica della DLL da utilizzare (riga 23) verrà fatto un tentativo di caricamento delle 3 librerie.

```
24.         For intConta = 4 To 6
25.             lngRis = LoadLibrary(Choose(intConta - 3, "STKIT432", "VB5STKIT",
    "VB6STKIT"))
26.             Call FreeLibrary(lngRis)
27.             If lngRis <> 0 Then intVersione = intConta
28.         Next intConta
29.     End If
30.
```

Alla riga 24 inizia il ciclo di ricerca: il valore minimo è il valore 4 mentre il massimo è il 6. L'elaborazione che effettuerà sarà pertanto di 3 cicli. Alla riga 25 viene richiamata la funzione API *LoadLibrary* per caricare la libreria il cui nome è dato dal risultato della funzione **Choose**, un'istruzione molto semplice ed altrettanto sconosciuta a molti.

Essa consente di estrarre un valore da un elenco di valori. L'elenco nel nostro caso sono i nomi delle 3 librerie: STKIT432, VB5STKIT e VB6STKIT. In base al primo parametro (*intConta - 3*) sarà estratto il valore corrispondente dall'elenco. Il risultato di tale elaborazione sarà il nome di una delle 3 librerie. Tale nome verrà quindi passato alla funzione *LoadLibrary*.

Se la libreria può essere caricata la variabile **lngRis** conterrà l'[handle](#) del modulo caricato; nel caso contrario **lngRis** conterrà il valore 0. Subito dopo il caricamento, la stessa libreria viene scaricata dalla memoria perché non utile al nostro scopo (riga 26).

Se la libreria viene caricata (ovvero il valore di lngRis è diverso da 0) sarà possibile per il programma utilizzare tale versione di VB. Pertanto la variabile **intVersione** sarà posta uguale alla variabile **intConta**.

All'uscita del ciclo la variabile **intVersione** conterrà la versione più alta a disposizione utilizzabile per creare il file di collegamento. Se la libreria VB6STKIT è presente sarà riportato il valore 6, altrimenti saranno riportati i valori 5 o 4 in base all'esistenza delle relative librerie.

```
31.     lngRis = 0
32.     If intVersione = VisualBasic6 Then
33.         lngRis = VB6CreateShellLink(strGruppo, strTitolo, strPath, "", True,
    "$ (Programs)")
34.     ElseIf intVersione = VisualBasic5 Then
35.         lngRis = VB5CreateShellLink(strGruppo, strTitolo, strPath, "")
36.     ElseIf intVersione = VisualBasic4 Then
37.         lngRis = VB4CreateShellLink(strGruppo, strTitolo, strPath, "")
38.     End If
39.     CreateShellLink = CBool(lngRis)
40.     Exit Function
41. Err_Handler:
42.     CreateShellLink = False
43. End Function
44.
```

Prima di procedere alla creazione del file di collegamento la variabile lngRis sarà azzerata allo scopo di poter riconoscere se la chiamata alla funzione API ha eseguito il suo scopo.

Alle righe 32-38 viene chiamata la funzione API in corrispondenza al valore di **intVersione**. *VB6CreateShellLink* per la versione 6, *VB5CreateShellLink* per il valore 5 e *VB4CreateShellLink* per la versione 4. Tutte e tre le chiamate restituiscono un valore nella variabile **lngRis**. Pertanto il valore di uscita dalla funzione dipenderà dal valore di **lngRis**. Il valore **False** indicherà che il collegamento non è stato creato.

Alle righe 41-42 abbiamo la semplicissima gestione degli errori, impostando come valore di ritorno della nostra funzione il valore **False** indicativo che il collegamento non è stato creato.

```
45. Private Sub Form_Load()
46.     AppPath = App.Path
47.     If Right$(AppPath, 1) <> "\" Then AppPath = AppPath & "\"
48.     AppPath = AppPath & App.EXEName & ".exe"
49.     txtTitoloCollegamento.Text = App.Title
50. End Sub
51.
```

La prima operazione che verrà eseguita al caricamento del form riguarda la costruzione del percorso del file eseguibile cui far puntare il collegamento. Alla riga 46 viene innanzitutto recuperato il percorso della cartella in cui è eseguito il progetto. Se tale cartella non è la radice del disco ovvero il suo percorso non termina con un "\", sarà necessario aggiungerlo alla fine.

Alla riga 48 viene finalmente recuperato il nome del file eseguibile e le tre stringhe vengono concatenate tra loro nella variabile **AppPath**.

Alla riga 49 viene inizializzato il contenuto della casella di testo **txtTitoloCollegamento** con il titolo del progetto in esecuzione. L'utente potrà tuttavia modificare tale valore a suo

piacimento.

```
52. Private Sub cmdCollegamentoMenuAvvio_Click()  
53.     Call CreateShellLink(VisualBasicUnknown, ".", txtTitoloCollegamento.Text,  
    AppPath)  
54. End Sub  
55.  
56. Private Sub cmdCollegamentoDesktop_Click()  
57.     Call CreateShellLink(VisualBasicUnknown, "..\..\Desktop",  
    txtTitoloCollegamento.Text, AppPath)  
58. End Sub
```

La creazione del file di collegamento si conclude con una semplicissima chiamata alla funzione **CreateShellLink**. Nel nostro esempio verrà utilizzato il rilevamento automatico della versione della libreria da utilizzare. Avremmo potuto fornire un valore qualunque dell'enumerazione **VersioneVB**; il vantaggio dell'utilizzare il rilevamento automatico (**VisualBasicUnknown**) sta nel fatto che il programma è in grado di funzionare anche nel caso che la libreria da utilizzare non sia installata nel sistema. Il programma infatti proverà a cercare le tre librerie e così il programma è in grado di adattarsi al sistema in cui viene installato. Naturalmente deve esistere almeno una delle tre librerie: **STKIT432.DLL**, **VB5STKIT.DLL** o **VB6STKIT.DLL**.

La riga 53 crea il collegamento nella cartella Programmi del Menu Avvio ed a tale scopo il percorso relativo della cartella sarà "." (come visto in precedenza).

Per creare invece il file di collegamento sul Desktop utilizzeremo il riferimento al percorso relativo "..\..\Desktop".

Gli altri parametri che forniremo saranno il titolo del collegamento, ovvero il valore contenuto nella casella di testo **txtTitoloCollegamento** ed il percorso del file eseguibile recuperato in precedenza e memorizzato nella variabile **AppPath**. Non forniremo invece nessun parametro extra al file eseguibile.

La dimostrazione del progetto è semplice quanto banale. Basterà inserire il titolo nella casella di testo e premere uno dei due pulsanti in funzione della locazione in cui si vuole creare il file di collegamento.

Resta da sciogliere solo un semplice dubbio: come creare un collegamento in una cartella differente? Infatti la funzione *fCreateShellLink* utilizza un percorso relativo alla cartella Programmi del Menu Avvio e potrebbe rendersi difficoltosa la costruzione di un percorso di un'altra cartella oppure di un altro disco differente da quello in cui si trova il Menu Avvio stesso.

La soluzione è altrettanto semplice: basterà creare il file di collegamento in una cartella conosciuta quale il Desktop o il Menu Avvio, recuperare il percorso assoluto di questa cartella e poi spostare il file dalla prima cartella a quella in cui desideriamo inserire il collegamento.

Potremmo ad esempio creare un collegamento sul Desktop e poi spostare (oppure copiare ed in seguito eliminare l'originale) il file di collegamento nella nostra cartella "D:\Documenti". Le istruzioni per spostare, copiare ed eliminare un file sono rispettivamente *Name*, *FileCopy* e *Kill*.

Ignorando le limitazioni della soluzione applicata e le eventuali complicazioni in caso di una cartella o di un disco differenti, il progetto si presenta davvero semplicissimo.

L'unico appunto che è doveroso fare riguarda la distribuzione della libreria utilizzata per la creazione del file di collegamento. Sarà infatti necessario assicurarsi che essa venga copiata nella cartella System o System32 del computer in cui viene utilizzata.

Con alcune versioni di Visual Basic 6 la documentazione allegata non specifica che la funzione *fCreateShellLink* della libreria VB6STKIT.DLL richiede due parametri aggiuntivi, anzi un esempio presente sul CD utilizza la stessa dichiarazione delle librerie precedenti. Si tratta ovviamente di una svista dello staff [Microsoft](#), corretta soltanto in seguito.

[Fibia FBI](#)

27 Febbraio 2002

Rivisto e corretto il 24 Marzo 2002



[Torna all'introduzione delle Richieste dei lettori](#)

---