

Connettere un'unità di rete

http://www.vbsimple.net/activity/act_20.htm

Richiesta di: [Diego Indaimo](#) - 14 Aprile 2002

Difficoltà: 4 / 5

Lavoro in un ufficio collegato tramite rete locale. Molto spesso sono costretto a collegarmi agli altri computer tramite il comando NET USE. Vorrei sapere come posso implementare questo comando in un applicazione scritta in Visual Basic.

Esistono svariate soluzioni per questo problema. La più semplice consiste nel richiamare lo stesso comando DOS tramite la funzione **Shell**. Ad esempio:

```
Call Shell("NET USE T: \\LINUX\tmp pass")
```

Richiama il comando **NET USE** indicando di connettere l'unità **T:** alla condivisione **tmp** del server **LINUX**, utilizzando la password **pass**.

In questo articolo invece dedicheremo una particolare cura allo sviluppo di un modulo per la scelta di una cartella di rete condivisa. Potrà quindi essere connessa un'unità di rete alla cartella condivisa selezionata. Questa soluzione ha molti punti in comune con l'[HowTo dedicato alla selezione di una cartella](#) ed utilizza la stessa funzione **ShBrowseForFolder** con un pizzico di personalizzazione in più. Prima di procedere si raccomanda la consultazione del citato articolo.

La personalizzazione consiste nell'utilizzo di una funzione di [Callback](#) che riceverà i [messaggi](#) inviati dalla finestra di dialogo e regolerà la possibilità di scegliere una cartella che sia condivisa piuttosto che un'altra che non lo sia. L'impostazione della funzione di callback è effettuato impostando il [membro lpfn](#) della struttura **BROWSEINFO**.

Il codice per il richiamo di tale finestra di dialogo deve necessariamente stare o utilizzare un modulo standard poiché la funzione **AddressOf** di Visual Basic, che restituisce un [puntatore](#) ad una funzione, può essere utilizzato soltanto facendo riferimento ad una funzione posta all'interno di un modulo standard. Vediamo pertanto il codice:

```
1. Option Explicit
2.
3. Private Type BROWSEINFO
4.     hOwner As Long
5.     pidlRoot As Long
6.     pszDisplayName As String
7.     lpszTitle As String
8.     ulFlags As Long
9.     lpfn As Long
10.    lParam As Long
11.    iImage As Long
12. End Type
13.
```

```

14. Private Const BIF_RETURNONLYFSDIRS = &H1
15. Private Const BIF_STATUSTEXT = &H4
16. Private Const CSIDL_NETWORK = &H12
17. Private Const MAX_PATH = 260
18. Private Const WM_USER = &H400
19. Private Const BFFM_SELCHANGED = 2
20. Private Const BFFM_ENABLEOK = (WM_USER + 101)
21. Private Const BFFM_SETSTATUSTEXTA = (WM_USER + 100)
22.

```

Il tipo di dati **BROWSEINFO** servirà per specificare una serie di dati utilizzati dalla funzione **ShBrowseForFolder**. Tali dati sono l'handle della finestra che richiama l'altra finestra di dialogo (*hOwner*), la cartella che farà da radice nella selezione (*pidlRoot*), il titolo che apparirà nella finestra di dialogo (*lpzTitle*), le opzioni di scelta (*ulFlags*), il puntatore alla funzione di Callback (*lpfn*) e degli eventuali dati aggiuntivi da passare alla funzione di Callback (*lParam*). Gli altri due membri (*pszDisplayName* ed *iImage*) non sono utilizzati in questo esempio.

Alle righe 14-21 seguono le dichiarazioni delle costanti [API](#): la prima è già stata spiegata nell'[altro articolo](#). **BIF_STATUSTEXT** consente di aggiungere un messaggio alla finestra di dialogo tramite la funzione di Callback. **CSIDL_NETWORK** definisce l'insieme delle Risorse di rete e verrà utilizzata da radice nella ricerca. **MAX_PATH** indicherà la lunghezza massima di un percorso di una cartella e sarà utilizzata per allocare un [buffer](#) sufficientemente ampio. Le costanti alle righe 19-21 indicano tre messaggi relativi alla finestra di dialogo e saranno spiegati più avanti.

```

23. Private Declare Function SHBrowseForFolder Lib "shell32.dll" Alias
    "SHBrowseForFolderA" (lpBrowseInfo As BROWSEINFO) As Long
24. Private Declare Function SHGetPathFromIDList Lib "shell32.dll" Alias
    "SHGetPathFromIDListA" (ByVal pidl As Long, ByVal pszPath As String) As Long
25. Private Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal pv As Long)
26. Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd
    As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
27.

```

Dovremmo già conoscere le tre funzioni dichiarate alle righe 23-25; ad esse aggiungiamo la ben nota **SendMessage** che consente di inviare un messaggio specifico ad una finestra ben definita.

```

28. Public Function SfoggiaCondivisione(ByVal hWnd As Long, ByRef strCondivisione As
    String, Optional ByVal strErrorStatus As String = "") As Boolean
29.     Dim BInfo As BROWSEINFO
30.     Dim lngPercorsoAllocato As Long
31.     Dim strPercorsoScelto As String
32.     With BInfo
33.         .hOwner = hWnd
34.         .lpfn = FunPtr(AddressOf BrowseCallbackProc)
35.         .lpzTitle = "Scegli una condivisione:"
36.         .pidlRoot = CSIDL_NETWORK
37.         .ulFlags = BIF_RETURNONLYFSDIRS
38.         If strErrorStatus <> "" Then
39.             .ulFlags = .ulFlags + BIF_STATUSTEXT
40.             strErrorStatus = StrConv(strErrorStatus, vbFromUnicode)
41.             .lParam = StrPtr(strErrorStatus)
42.         End If
43.     End With

```

La funzione **SfoggiaCondivisione** si occuperà di mostrare all'utente la finestra di dialogo per la scelta della cartella condivisa. Essa richiede l'[handle](#) del form che la richiama, una variabile buffer di nome **strCondivisione** per contenere il percorso della cartella recuperata

ed infine consente opzionalmente di fornire una stringa che verrà mostrata nel momento in cui l'utente seleziona una cartella non condivisa (**strErrorStatus**).

Alle righe 32-43 verrà riempita la struttura **BInfo** di tipo **BROWSEINFO**. La riga 33 specifica l'handle della finestra, mentre la successiva assegna al membro **lpfn** il puntatore alla funzione di Callback tramite la funzione **FunPtr** che vedremo dopo.

Verrà impostata la radice delle cartelle ed anche il titolo della finestra di dialogo (righe 35 e 36). Se è stato specificato un messaggio di errore per la scelta delle cartelle condivise sarà allora necessario specificare nelle opzioni di utilizzare tale messaggio quando richiesto dalla funzione di Callback. La stringa da passare verrà prima convertita in formato **ANSI** e ne verrà passato il **puntatore** nel membro **lParam** (righe 38-42).

```

44. lngPercorsoAllocato = SHBrowseForFolder(BInfo)
45. strPercorsoScelto = Space$(MAX_PATH)
46. SfoggiaCondivisione = SHGetPathFromIDLList(lngPercorsoAllocato, strPercorsoScelto)
47. If SfoggiaCondivisione = True Then
48.     CoTaskMemFree lngPercorsoAllocato
49.     strCondivisione = Left$(strPercorsoScelto, InStr(strPercorsoScelto, Chr$(0)) -
1)
50. End If
51. End Function
52.
```

Alla riga 44 verrà richiamata la finestra di dialogo con le opzioni specificate in precedenza. Il procedimento di estrazione della cartella selezionata è pressoché identico a quello specificato nell'altro articolo. La cartella selezionata sarà quindi memorizzata nella variabile di ritorno **strCondivisione**.

```

53. Private Function FunPtr(ByVal lngFn As Long) As Long
54.     FunPtr = lngFn
55. End Function
56.
```

La funzione **FunPtr** è estremamente semplice, tanto da sembrare inutile. Purtroppo Visual Basic non consente di assegnare il valore di un'espressione **AddressOf** ad una variabile. È pertanto necessario passare tale espressione ad una funzione fantasma che restituirà come valore di ritorno il puntatore fornito (riga 54).

```

57. Private Function BrowseCallbackProc(ByVal hWnd As Long, ByVal uMsg As Long, ByVal
lParam As Long, ByVal lpData As Long) As Long
58.     Dim strBuf As String
59.     Dim intPos As Integer
60.     If uMsg = BFFM_SELCHANGED Then
61.         strBuf = Space$(MAX_PATH)
62.         If SHGetPathFromIDLList(lParam, strBuf) Then
```

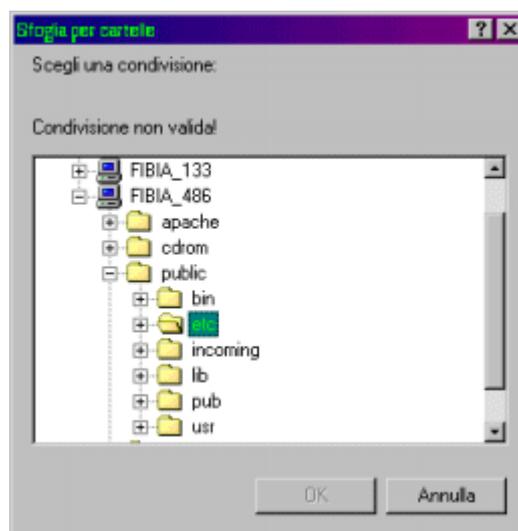
La funzione di Callback prende il nome di **BrowseCallbackProc** e consente di ricevere i messaggi inviati dalla finestra di dialogo. L'unico messaggio che interessa noi è **BFFM_SELCHANGED** (riga 60), che indica che la selezione si è spostata su un elemento della lista. L'elemento selezionato si trova in **lParam**, sarà estratto alle righe 61 e 62 e dovrà essere controllato.

Se il percorso della cartella scelta è una cartella condivisa conterrà due backslash all'inizio (\\), il nome del computer, un ulteriore barra rovesciata ed il nome della cartella condivisa. Un esempio potrebbe essere \\LINUX\tmp.

La presenza di ulteriori backslash indicherà una sottocartella della cartella condivisa selezionata. Tali sottocartelle saranno evitate bloccando il pulsante OK che consente la chiusura della finestra di dialogo.

Il blocco del pulsante OK è effettuato inviando il messaggio **BFFM_ENABLEOK** alla finestra di dialogo e specificando nell'argomento **lParam** un valore 0. Nel caso venga selezionata una cartella non condivisa sarà anche mostrato l'eventuale messaggio di errore passato alla struttura alla riga 41.

Nella figura fianco è stata selezionata la cartella **etc** della condivisione **public** sul computer **FIBIA_486**. Non essendo una cartella condivisa (ma una sua sottocartella) il pulsante OK è stato bloccato ed in alto è apparso l'avviso *Condivisione non valida*.



```

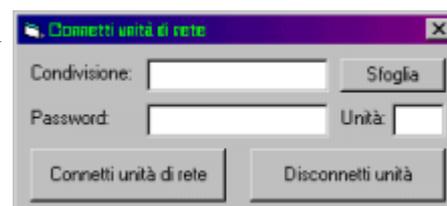
63.     intPos = InStr(3, strBuf, "\")
64.     If InStr(intPos + 1, strBuf, "\") > 0 Then
65.         SendMessage hWnd, BFFM_ENABLEOK, ByVal 0&, ByVal 0&
66.         SendMessage hWnd, BFFM_SETSTATUSTEXTA, ByVal 0&, ByVal lpData
67.     Else
68.         SendMessage hWnd, BFFM_SETSTATUSTEXTA, ByVal 0&, ByVal vbNullChar
69.     End If
70. Else
71.     SendMessage hWnd, BFFM_SETSTATUSTEXTA, ByVal 0&, ByVal lpData
72.     SendMessage hWnd, BFFM_ENABLEOK, ByVal 0&, ByVal 0&
73. End If
74. End If
75. End Function

```

L'accennato controllo è svolto alle righe 63 e 64. Se non è presente la barra di separazione tra il nome del computer e la cartella, sarà segno che la selezione si trova sopra il nome di un computer e pertanto (righe 70-72) sarà bloccato il pulsante OK della finestra di dialogo e verrà mostrato il messaggio di errore. Se la barra di separazione tra nome del computer e cartella è presente, sarà quindi verificata la presenza di un'ulteriore barra (per una sottocartella). La sua presenza bloccherà il pulsante OK e mostrerà il messaggio di errore (righe 65 e 66).

In caso di assenza di quest'ultima condizione (ovvero di un'ulteriore barra dopo quella che separa il nome del computer dalla condivisione), la cartella scelta non è né il nome di un computer né una sottocartella di una cartella condivisa. È invece una cartella condivisa e pertanto sarà cancellato l'eventuale avviso passato dagli errori precedenti (riga 71).

Possiamo provare il modulo appena sviluppato con il form presentato a fianco, costituito da una serie di *Label*, *TextBox* e *CommandButton*. I controlli fondamentali sono la casella di testo **txtCondivisione**, il pulsante **cmdSfoggia**, le caselle **txtPassword** (con proprietà **PasswordChar** impostata su *) e **txtDisco** (con proprietà **MaxLength** impostata su 1) ed infine i due pulsanti in basso **cmdConnetti** e **cmdDisconnetti**. Il funzionamento è molto semplice: l'utente potrà immettere



manualmente il percorso della cartella condivisa oppure effettuare la scelta premendo il pulsante **Sfoglia**. Dovrà anche immettere l'eventuale password e specificare la lettera dell'unità da assegnare, prima di premere il pulsante **Connetti unità di rete**. Per disconnettere un'unità sarà invece necessaria soltanto la lettera dell'unità e la pressione del pulsante **Disconnetti unità**.

Il codice del form si compone di sole tre procedure legate al click sopra ognuno dei tre pulsanti:

```

1. Option Explicit
2.
3. Private Declare Function WNetAddConnection Lib "mpr.dll" Alias
   "WNetAddConnectionA" (ByVal lpszNetPath As String, ByVal lpszPassword As String,
   ByVal lpszLocalName As String) As Long
4. Private Declare Function WNetCancelConnection Lib "mpr.dll" Alias
   "WNetCancelConnectionA" (ByVal lpszName As String, ByVal bForce As Long) As Long
5.
6. Private Sub cmdSfoglia_Click()
7.     Dim strCondivisione As String
8.     If SfogliaCondivisione(Me.hWnd, strCondivisione, "Condivisione non valida!") Then
9.         txtCondivisione.Text = strCondivisione
10.    End If
11. End Sub
12.

```

Alla riga 3 è dichiarata la funzione [API WNetAddConnection](#) che consente la connessione di un'unità rete, mentre la funzione successiva **WNetCancelConnection** consente la sua disconnessione.

Alla riga 8 è richiamata la funzione SfogliaCondivisione dal modulo, fornendole l'[handle](#) del form che richiama la finestra di dialogo, un [buffer](#) di nome **strCondivisione** ed il messaggio di errore **"Condivisione non valida!"** sulle cartelle non condivise. Se la funzione ritorna un valore True sarà segno che l'utente ha effettuato la sua scelta premendo il pulsante OK. Sarà pertanto inserito il percorso della cartella scelta nella casella **txtCondivisione** (riga 9).

```

13. Private Sub cmdConnetti_Click()
14.     If WNetAddConnection(txtCondivisione.Text, txtPassword.Text, txtDisco & ":") Then
15.         MsgBox "Errore durante la connessione a " & txtCondivisione.Text, vbCritical +
   vbOKOnly, Me.Caption
16.     Else
17.         MsgBox "Unità di rete " & txtDisco.Text & ": connessa!", vbInformation +
   vbOKOnly, Me.Caption
18.     End If
19. End Sub
20.

```

Alla pressione del pulsante **cmdConnetti** verrà connessa l'unità di rete tramite la funzione **WNetAddConnection**. In ambo i casi di successo o fallimento sarà mostrata una finestra di avviso con le relative informazioni.

```

21. Private Sub cmdDisconnetti_Click()
22.     If WNetCancelConnection(txtDisco & ":", ByVal 0&) Then
23.         MsgBox "Errore durante la disconnessione di " & txtDisco & ":", vbCritical +
   vbOKOnly, Me.Caption
24.     Else
25.         MsgBox "Unità di rete " & txtDisco.Text & ": disconnessa!", vbInformation +
   vbOKOnly, Me.Caption
26.     End If
27. End Sub

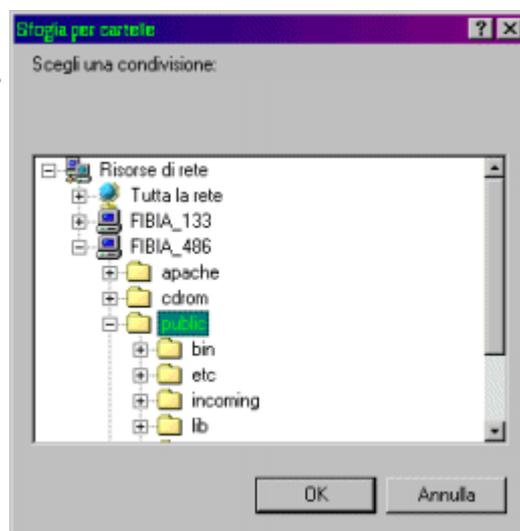
```

Alla stessa maniera, alla pressione del pulsante **cmdDisconnetti** dovrà essere disconnessa l'unità di rete tramite la funzione **WNetCancelConnection** ed in base al risultato dell'operazione sarà mostrato un avviso.

La dimostrazione del funzionamento del progetto è molto semplice. Basterà cliccare sul pulsante *Sfoglia*, selezionare una cartella condivisa, notando come il pulsante OK si attiva e si disattiva in funzione della selezione; si comporta allo stesso modo anche il messaggio informativo tra il titolo e l'elenco delle cartelle.

Nel nostro esempio abbiamo scelto la condivisione **public** sul computer **FIBIA_486**.

Alla pressione del pulsante OK sarà ricopiato il percorso della cartella nella casella di testo **txtCondivisione**.



Non ci resta che riempire i dati mancanti fornendo l'eventuale password per la cartella e la lettera dell'unità da assegnare e premere il pulsante *Connetti unità di rete*.

Oltre che con il messaggio informativo possiamo verificare l'avvenuta connessione aprendo la finestra *Risorse del computer* per vedere l'unità di rete connessa, come mostrato nella figura sotto.



Figura 4

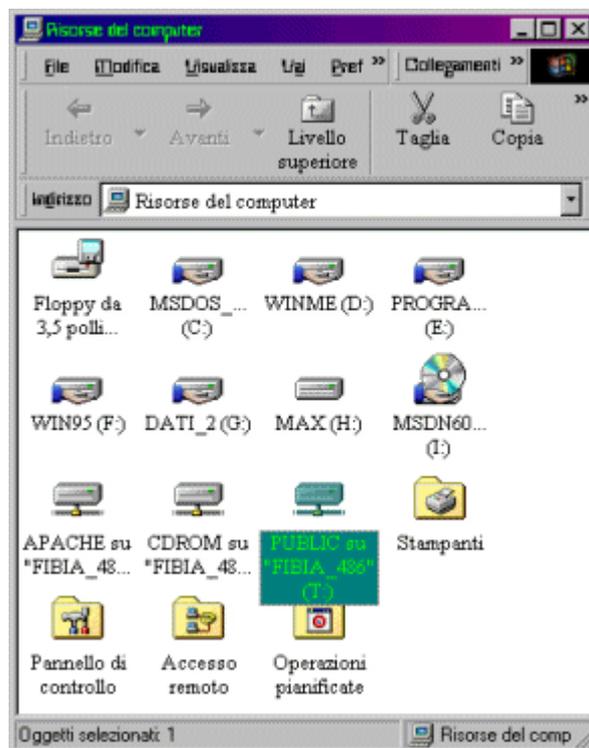


Figura 5

Questo articolo implementa una soluzione per la connessione di unità di rete molto comoda ed utilizza una tecnica di Callback molto semplice. È stata provata utilizzando una rete di tipo Microsoft ed unità Linux connessa tramite Samba nel formato di condivisione Windows NT.

Non si pongono rischi particolari connessi all'utilizzo di questa soluzione in quanto tutte le operazioni di richiamo della funzione di Callback sono svolte dalla funzione API. Potrebbero invece presentarsi problemi se la funzione di Callback viene modificata in maniera impropria.

[Fibia FBI](#)
22 Aprile 2002



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