
[Home Page](#)
[Novità](#)
[Aiuto](#)

Seleziona un file per l'apertura o il salvataggio

http://www.vbsimple.net/howto/ht_055.htm
Difficoltà: 3 / 5

L'insieme dei controlli [Thunder](#) contiene tre controlli dedicati alla scelta di uno o più files su richiesta dell'utente; si tratta dei controlli *DriveListBox* , *DirListBox* e *FileListBox* , tre controlli decisamente troppo antiquati per continuare a comparire anche nel più banale progetto per Windows.

Ogni volta, infatti, che all'utente è richiesto di selezionare un file per l'apertura o il salvataggio, le applicazioni Windows presentano una familiare finestra di dialogo comune; è possibile richiamare questa finestra utilizzando il controllo [OCX Microsoft Common Dialog](#) oppure utilizzando l'[API](#), risparmiando quindi, la distribuzione del controllo aggiuntivo. In quest'articolo sarà affrontata la seconda delle soluzioni, in quanto la prima, data la sua semplicità, non richiede particolari spiegazioni.

Data la generalità d'uso delle due funzioni si è preferito inserirle all'interno di un modulo standard da incorporare all'interno del progetto che richiedesse la funzionalità. Il modulo si apre con la dichiarazione delle costanti e delle funzioni API:

```

1. Option Explicit
2.
3. Private Type OPENFILENAME
4.     lStructSize As Long
5.     hwndOwner As Long
6.     hInstance As Long
7.     lpstrFilter As String
8.     lpstrCustomFilter As String
9.     nMaxCustFilter As Long
10.    nFilterIndex As Long
11.    lpstrFile As String
12.    nMaxFile As Long
13.    lpstrFileName As String
14.    nMaxFileName As Long
15.    lpstrInitialDir As String
16.    lpstrTitle As String
17.    flags As Long
18.    nFileOffset As Integer
19.    nFileExtension As Integer
20.    lpstrDefExt As String
21.    lCustData As Long
22.    lpfnHook As Long
23.    lpTemplateName As String
24. End Type
25.

```

La struttura **OPENFILENAME** verrà utilizzata da entrambe le funzioni di richiesta di apertura e di salvataggio e richiede la specifica dei filtri sui nomi dei files visualizzati, del titolo della finestra, della cartella iniziale presentata e la scelta di una serie di flags che determineranno il comportamento della finestra. Per brevità non saranno presentati tutti i flags ma solo quelli utilizzati in questa soluzione:

```

26. Private Const OFN_FILEMUSTEXIST As Long = &H1000
27. Private Const OFN_HIDEREADONLY As Long = &H4
28.

```

Le due costanti specificheranno di verificare l'esistenza del file scelto e di nascondere la casella di controllo *Apri in sola lettura*, durante l'operazione di apertura.

```

29. Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias
    "GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long
30. Private Declare Function GetSaveFileName Lib "comdlg32.dll" Alias
    "GetSaveFileNameA" (pOpenfilename As OPENFILENAME) As Long
31.

```

Le due funzioni *GetOpenFileName* e *GetSaveFileName* servono rispettivamente per l'operazione di apertura e di salvataggio su file. Entrambe utilizzano una variabile di tipo **OPENFILENAME** e si differenziano solo per pochissime differenze.

```

32. Private OFName As OPENFILENAME
33.



```

La variabile **OFNAME** sarà utilizzata da entrambe le funzioni di apertura e salvataggio:

```

34. Public Function ShowOpen(FormHandler As Form, ByVal Filtro As String) As String
35.     With OFName
36.         .lStructSize = Len(OFName)
37.         .hwndOwner = FormHandler.hwnd
38.         .hInstance = App.hInstance
39.         .lpstrFilter = Filtro & vbNullChar
40.         .lpstrFile = Space$(254)
41.         .nMaxFile = 255
42.         .lpstrFileTitle = Space$(254)
43.         .nMaxFileTitle = 255
44.         .lpstrInitialDir = CurDir
45.         .lpstrTitle = "Seleziona il file da aprire"
46.         .flags = OFN_FILEMUSTEXIST Or OFN_HIDEREADONLY
47.     End With
48.     If GetOpenFileName(OFName) Then ShowOpen = Trim$(OFName.lpstrFile)
49. End Function
50.

```

La funzione **ShowOpen** utilizza lo stesso nome del metodo  del controllo *Common Dialog*  ma in aggiunta richiede il passaggio del form per il quale la finestra di dialogo sarà richiamata ed una stringa contenente il filtro da applicare alla visualizzazione. Il filtro utilizza una forma particolare che verrà affrontata più avanti.

La funzione **ShowOpen** si compone di due soli passaggi: riempimento della variabile **OFNAME** (righe 35-47) e richiamo della funzione *GetOpenFileName*.

Alla riga 36 è definita la dimensione della struttura mediante la funzione *Len*; il form fornito nel richiamo della funzione (**FormHandler**) sarà utilizzato solo per recuperare l'[handle](#) per il quale la finestra di dialogo diverrà modale; alla riga 39 sarà assegnato il filtro per la visualizzazione dei files; la riga 44 invece determinerà quale cartella sarà mostrata all'apertura della finestra e nel nostro esempio corrisponde alla cartella corrente (funzione *CurDir*); il titolo della finestra sarà **"Seleziona il file da aprire"**.

Nell'ultima assegnazione saranno impostati i flags di visualizzazione; in questo esempio ne sono stati utilizzati soltanto due: **OFN_FILEMUSTEXIST** e **OFN_HIDEREADONLY**, ma all'occorrenza possono essere combinati altri flags mediante l'operatore **Or**.

Alla riga 48 è richiamata la funzione *GetOpenFileName* con la struttura **OFNAME** e nel caso che la funzione restituisca un valore diverso da zero, sarà possibile leggere il nome del file selezionato dalla struttura **OFNAME**.

La funzione **ShowSave** opera in maniera analoga alla precedente salvo che per l'impostazione dei flags, qui non necessari:

```
51. Public Function ShowSave(FormHandler As Form, ByVal Filtro As String) As String
52.     With OFName
53.         .lStructSize = Len(OFName)
54.         .hwndOwner = FormHandler.hwnd
55.         .hInstance = App.hInstance
56.         .lpstrFilter = Filtro & vbNullChar
57.         .lpstrFile = Space$(254)
58.         .nMaxFile = 255
59.         .lpstrFileName = Space$(254)
60.         .nMaxFileName = 255
61.         .lpstrInitialDir = CurDir
62.         .lpstrTitle = "Seleziona il file in cui salvare"
63.         .flags = 0
64.     End With
65.     If GetSaveFileName(OFName) Then ShowSave = Trim$(OFName.lpstrFile)
66. End Function
```

Come già accennato la differenza si limita alla riga 63 poiché non saranno necessari particolari controlli durante il salvataggio del file; la riga 65 inoltre richiama la funzione *GetSaveFileName* anziché la precedente.

L'utilizzo del modulo appena sviluppato è davvero molto semplice: porremo sopra un form una casella di testo di nome **txtFileName** e due pulsanti di nome **cmdOpen** e **cmdSave**.



La pressione del pulsante **Apri** mostrerà la relativa finestra di dialogo per l'apertura, mentre la pressione del pulsante **Salva** mostrerà la finestra di salvataggio.

```
1. Option Explicit
2.
3. Private Sub cmdOpen_Click()
4.     txtFileName.Text = ShowOpen(Me, _
5.         "Files di testo (*.txt)" & vbNullChar & "*.txt")
6. End Sub
7.
8. Private Sub cmdSave_Click()
9.     txtFileName.Text = ShowSave(Me, _
10.        "Files di testo (*.txt)" & vbNullChar & "*.txt" & vbNullChar & _
11.        "Tutti i files (*.*)" & vbNullChar & "*.*)"
12. End Sub
```

Il richiamo delle due funzioni precedentemente esposte richiede il passaggio del form per il quale la finestra diverrà modale e il filtro di visualizzazione; le due funzioni si differenziano anche per il filtro utilizzato: la prima delle due funzioni utilizzerà un solo filtro corrispondente ai files di testo (*.txt), mentre la seconda funzione utilizzerà due differenti filtri, corrispondenti ai files di testo (*.txt) ed a tutti i files (*.*)).

Ogni filtro si compone di due parti: il testo visibile ed il filtro vero e proprio, separati tra

loro dal codice [ASCII](#) 0; facendo riferimento al primo esempio il filtro si compone della descrizione "*Files di testo (*.txt)*" e del filtro "**.txt*".

Per separare un filtro dal successivo si utilizza lo stesso codice ASCII 0, come segue:

```
[Descrizione][0][Filtro][0][Descrizione2][0][Filtro2]
```

dove naturalmente [0] corrisponde al codice ASCII 0, indicato anche come costante `vbNullChar`.

Nell'esempio alle righe 9-11 sono assegnati due filtri: "*Files di testo (*.txt)*" e "*Tutti i files (*.*)*" e la differenza può dimostrarsi eseguendo il progetto.

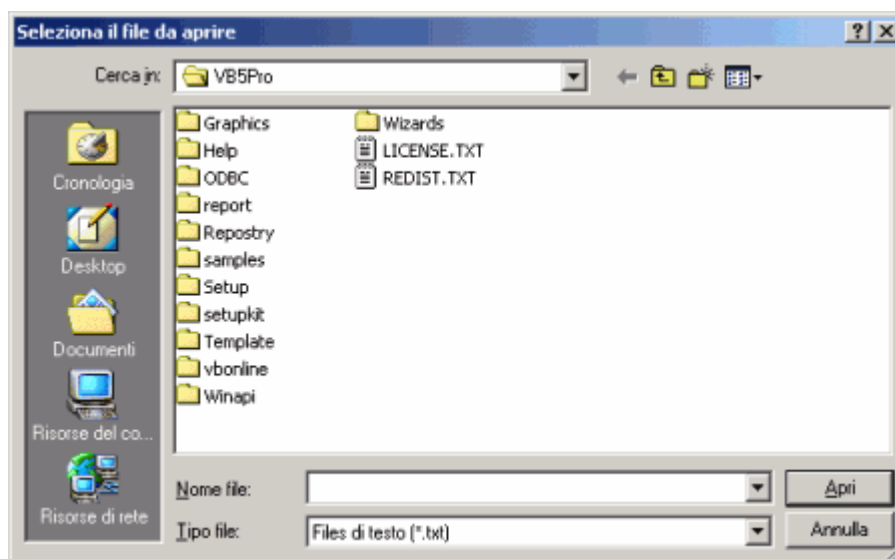


Figura 2

La pressione del pulsante **Apri** mostrerà una finestra simile a quella mostrata nella figura 2 (lo stile della stessa dipende dalla versione di Windows installata); la casella in basso **Tipo file** specifica i filtri nella visualizzazione dei files. Naturalmente saranno visualizzati solo i files corrispondenti al filtro selezionato.

Alla pressione del pulsante **Apri** sarà assegnato solo un filtro, per cui non sarà possibile variarlo tramite la ComboBox sul

fondo. Nel caso invece del pulsante **Salva**, saranno assegnati due filtri e sarà possibile selezionare quello più idoneo, come mostrato nella figura qui mostrata.



Tutte le moderne applicazioni Windows dovrebbero evitare l'uso dei controlli *DriveListBox*, *DirListBox* e *FileListBox* perché troppo antiquati rispetto l'evoluzione dei sistemi Windows. L'uso del controllo *Microsoft Common Dialog* può dare un aspetto molto più moderno al progetto, ma richiede l'uso di un controllo esterno aggiuntivo.

Il modulo qui presentato riproduce esattamente il comportamento dei metodi *ShowOpen* e *ShowSave* del controllo di casa [Microsoft](#), rendendo quindi superfluo l'uso di quel controllo se utilizzato solo per i due metodi dedicati alla scelta di un file.



[Torna all'indice degli HowTo](#)
