





Leggere il contenuto di un file

http://www.vbsimple.net/howto/ht_003.htm


Difficoltà:   2 / 5

La lettura di un file di testo è un'operazione semplicissima in Visual Basic. Vedremo in quest'esempio come visualizzare il contenuto di due files di testo, **FILE1.TXT** e **FILE2.TXT** posti all'interno della cartella del programma.

All'interno di un form inseriamo due pulsanti di comando  entrambi di nome **PulsanteLeggi**. Formeremo quindi una matrice di pulsanti. Il primo avrà indice 0 ed il secondo indice 1.

Inseriamo anche una casella di testo  e chiamiamola **CasellaTesto**. Impostiamo la sua proprietà *Multiline* a True permettendo quindi il ritorno a capo del testo, e la *ScrollBars* a 3 - Both, inserendo le barre di scorrimento in entrambe le direzioni. Conviene anche impostare la proprietà *Locked* a True in modo da impedire la modifica del testo letto.

Abbandoniamo l'interfaccia grafica e focalizziamoci sul codice.

Per comodità dichiareremo due [costanti](#) : *FileDaAprire1* e *FileDaAprire2*. La prima conterrà il nome del primo file da aprire, ovvero FILE1.TXT e la seconda conterrà il nome dell'altro file da aprire (FILE2.TXT).

```
1. Private Const FileDaAprire1 = "FILE1.TXT"
2. Private Const FileDaAprire2 = "FILE2.TXT"
3.
```

Segue la funzione legata all'evento Click sopra uno dei due pulsanti. Poiché gli elementi sono in matrice, entrambi rispondono allo stesso evento. Si differenziano tra loro per il parametro Index passato.

Ecco il codice completo che analizzeremo passo dopo passo:

```
4. Private Sub PulsanteLeggi_Click(Index As Integer)
5.     Dim BUFFER As String
6.     Dim NRFILE As Integer
7.     CasellaTesto.Text = ""
8.     NRFILE = FreeFile
9.     If Index = 0 Then
10.        Open App.Path & "\" & FileDaAprire1 For Binary As NRFILE
11.     Else
12.        Open App.Path & "\" & FileDaAprire2 For Binary As NRFILE
13.     End If
14.     If LOF(NRFILE) > 20000 Then MsgBox "Il file é più grande di 20KB. Si potrebbe
verificare un errore", vbCritical
15.     While Not EOF(NRFILE)
16.         BUFFER = Space(2048)
17.         Get NRFILE, , BUFFER
18.         BUFFER = CasellaTesto.Text & BUFFER
19.         CasellaTesto.Text = BUFFER
20.     Wend
21.     Close NRFILE
22. End Sub
```

Alle righe di codice 5 e 6 dichiariamo due variabili: **BUFFER**, utilizzata per immagazzinare i dati che provengono dalla lettura del file, e **NRFILE**, un numero che identifica il file aperto.

La prima operazione di questa routine consiste nell'azzerare il contenuto precedente della casella di testo, con il semplice assegnamento di una stringa nulla alla proprietà *Text* di **CasellaTesto**.

Segue, alla riga 8, la ricerca di un numero di file libero. Infatti tutti i files aperti con Visual Basic richiedono un numero di file. L'istruzione **FreeFile** restituisce il prossimo numero di file libero.

Subito segue un semplice controllo del parametro Index. Se esso è 0, ovvero è stato cliccato il primo pulsante, effettua l'apertura del primo file, altrimenti apre il secondo file.

L'istruzione **Open** che abbiamo alla settima e alla nona riga ha questa sintassi:

Open NOMEFILE **For** MODALITÀ **As** NUMEROFILE

NOMEFILE ovviamente indica il percorso completo del file da aprire.

MODALITÀ indica l'utilizzo che vogliamo effettuare con il file.

NUMEROFILE indica un numero che identificherà univocamente il file da aprire.

La modalità di apertura cambia il modo di recupero dei dati dal file aperto. Il parametro MODALITÀ può assumere uno di questi valori:

- APPEND - È possibile soltanto aggiungere dati alla fine del file.
- BINARY - Lettura e scrittura in maniera binaria.
- INPUT - Lettura dal file.
- OUTPUT - Scrittura del file.
- RANDOM - Modalità casuale.

Le nostre istruzioni alle righe 10 e 12 innanzitutto ricavano il nome del file da aprire.

Abbiamo detto che le costanti **FileDaAprire1** e **FileDaAprire2** contengono il nome del file da aprire, ma non il suo percorso fisico. Sappiamo che entrambi i files si trovano all'interno della cartella del programma.

Effettueremo quindi una ricostruzione del nome del file: *App.Path* indica il percorso della cartella del programma. A questo aggiungiamo una barra di separazione di directory ed infine il nome del file, attraverso le due costanti viste in precedenza.

Conoscendo quindi il percorso, l'istruzione effettua un'apertura in modalità **Binary** utilizzando il numero del file contenuto nella variabile **NRFile**, che in effetti è il primo numero libero utilizzabile per aprire un file.

Alla riga 14 abbiamo un controllo d'obbligo: poiché ci accingiamo a visualizzare il contenuto del file all'interno di una TextBox è fondamentale considerare che se il testo complessivo supera i 20 KB quasi sicuramente si genererà un errore.

Nel nostro progetto abbiamo inserito due files: uno di dimensione piccola ed un altro di dimensione maggiore di 20 KB.

Il controllo che viene effettuato è tramite l'istruzione *LOF* ovvero Length Of File, che riporta la dimensione del file aperto in bytes. Se tale numero è maggiore di 20000 bytes, avverte l'utente con un avviso, tramite l'istruzione *MsgBox*.

Segue un ciclo di lettura che si ripeterà fin quando non verrà letto l'intero file:

```
15. While Not EOF(NRFILE)
16.     BUFFER = Space(2048)
17.     Get NRFILE, , BUFFER
18.     BUFFER = CasellaTesto.Text & BUFFER
19.     CasellaTesto.Text = BUFFER
20. Wend
```

L'istruzione *EOF* - End Of File - controlla se la posizione di lettura ha raggiunto la fine del file. Per cui, fintanto che tale posizione non è raggiunta verranno eseguite le righe 16, 17, 18 e 19.

La prima di queste righe prepara un buffer di lettura di 2 KB.

Fatto questo possiamo effettuare la vera lettura del file. L'istruzione *Get* è quella che si occupa di questa funzione. Essa richiede un numero di file ed il nome di una variabile dove memorizzare i dati (BUFFER).

Letto il buffer di 2KB aggiungiamo all'inizio d'esso il contenuto attuale della casella di testo, contenuto che verrà aggiornato alla riga successiva. Le righe 18 e 19 camminano di pari passo. La seconda riceve il contenuto dalla prima e la prima nel passaggio successivo aggiunge il contenuto del buffer al contenuto precedente.

Lo svantaggio principale di questo genere di operazioni è che per ogni istruzione vengono passate grosse quantità di dati. I 2KB di buffer alla seconda operazione diventano 4, poi 6, etc...

Alla riga 21 segue un'istruzione d'obbligo. La chiusura del file. Infatti tutti i files aperti tramite l'istruzione *Open* rimangono bloccati fintanto che non viene chiuso il numero del file associato ad essi.

Il programmino è finito. L'utilizzo è estremamente facile. Il click sopra uno dei due pulsanti effettua la lettura del file corrispondente.

[Fibia FBI](#)

22 Novembre 2000



[Torna all'indice degli HowTo](#)
