Verificare se un file esiste

http://www.vbsimple.net/howto/ht_011.htm

- Primo sistema (Dir)
- Secondo sistema (Dir con controllo)
- Terzo sistema (GetAttr)
- Quarto sistema (Open)
- Testiamo le quattro funzioni

In molti programmi si presenta la necessità di verificare se un dato file esiste. Infatti, in tutti i programmi dove si utilizzano files, il tentativo di leggere dati da un file inesistente genera un errore che, se non correttamente gestito, provoca la chiusura immediata del programma.

Vedremo vari esempi per verificare l'esistenza di un file specifico, ognuno con i suoi vantaggi, svantaggi e limiti.

Primo sistema (Dir)

Difficoltà: 1 / 5

Probabilmente questo è il metodo più utilizzato e più semplice.
Consiste nell'utilizzare la funzione *Dir* che riporta tutti i files che corrispondono ad un certo criterio.

```
1. Public Function FileEsiste1(ByVal FileDaControllare As String) As Boolean
2.     Dim strVerifica As String
3.     On Error Resume Next
4.     FileEsiste1 = False
5.     strVerifica = Dir$(FileDaControllare)
6.     If (strVerifica <> "") And (Err.Number = 0) Then FileEsiste1 = True
7. End Function
```

La prima funzione si chiamerà **FileEsiste1** e richiede che sia passato come parametro il nome del file da controllare. All'uscita restituisce un valore booleano: True se il file cercato esiste e False se non esiste.

All'interno d'essa è dichiarata una variabile di nome **strVerifica** che conterrà il valore restituito dalla funzione Dir. In questo esempio abbiamo usato la corrispondente funzione *Dir$* che restituisce in uscita un valore di tipo String anziché di tipo Variant.
Alla riga 3 abbiamo inserito un'istruzione per la gestione degli errori. Questo genere di controlli è obbligatorio, poiché se l'utente chiede di controllare un file in un disco che non esiste (per esempio chiede un file da A: ma il dischetto non è inserito) oppure effettua una ricerca in una forma sbagliata, viene generato un errore. Senza questa riga di gestione degli errori il programma si interromperebbe mostrando una schermata di errore.

Subito, alla riga 4, prepara una risposta alla ricerca del file, impostando il contenuto di *FileEsiste1* a False. Questo perché il valore di risposta sarà True **solo** quando verrà rilevata l'esistenza del file.

Alla riga 5 abbiamo il controllo vero e proprio.
Abbiamo chiamato la funzione *Dir* con il nome del file da cercare. Se il file esiste essa ci restituirà il suo nome Windows, ovvero se il nome del file cercato è breve, ci verrà restituito il suo nome lungo. Se il file non esiste ci verrà restituita una stringa vuota.

Se, invece, viene inviata una richiesta errata, specificando una lettera di drive dove non è presente un disco, viene generato l'errore di runtime 52 (*Nome o numero di file non valido*). Ma, poiché alla riga 3 abbiamo inserito un'istruzione di gestione degli errori, non ci viene presentato alcun messaggio di errore ed il codice continua il suo ciclo.

Alla riga 6 abbiamo il controllo decisivo: viene innanzitutto controllato che il contenuto della variabile **strVerifica** non sia nullo e verifica inoltre che non sia avvenuto un errore.

In ogni caso, se il valore di Err.Number è diverso da zero, la funzione *FileEsiste1* restituisce il valore False, impostato alla riga 4. Questo perché il verificarsi di un errore presuppone un'errata richiesta di file.

Se invece il valore di Err.Number è 0 (nessun errore), l'esistenza del file deve essere comprovata dal valore di **strVerifica**. Se il file non esiste il suo contenuto sarà una stringa nulla; invece se il contenuto di **strVerifica** fosse non nullo avremmo la quasi sicurezza che il file esiste, ed in questo caso verrebbe eseguita l'istruzione che imposta il valore di *FileEsiste1* a True.

Lo svantaggio principale di questa funzione di controllo è che essa non è capace di distinguere un file da un nome di drive. Se, per errore, chiedessimo se esiste il file "C:\" essa ci risponderebbe che esiste; tuttavia non potremmo mai utilizzare la radice come se fosse un file. Inoltre se nella richiesta viene inserito un carattere jolly (? oppure *), ad esempio **"C:\Command.*"** la funzione *Dir* tratterebbe tale carattere alla stessa maniera di Windows, ovvero andrebbe a cercare tutti i files il cui nome sia Command, senza badare all'estensione. Tutto ciò potrebbe sembrare corretto: esiste almeno un file con quel nome, ma che succederebbe se provassimo ad aprire il file "C:\Command.*" ? Si genererebbe un errore che bloccherebbe il programma.

In definitiva il sistema di Dir è molto comodo, ma richiede che i dati inviati alla funzione siano prima trattati e controllati, al fine di evitare errori.

Secondo sistema (Dir con controllo)

Difficoltà: 2 / 5

Prendendo spunto dal sistema precedente, proviamo a migliorare il controllo evitando che vengano utilizzati i caratteri jolly (wildcards) nella specifica del nome del file.

```
1. Public Function FileEsiste2(ByVal FileDaControllare As String) As Boolean
```



```
2.      Dim strVerifica As String
3.      On Error Resume Next
4.      FileEsiste2 = False
5.
```

La funzione **FileEsiste2** è molto simile alla precedente, ma stavolta abbiamo inserito una piccola funzione di gestione degli errori. Al verificarsi di un errore il flusso del programma procede alla riga successiva.

```
6.      FileDaControllare = Trim(FileDaControllare)
7.      If InStr(1, FileDaControllare, "*") > 0 Then Exit Function
8.      If InStr(1, FileDaControllare, "?") > 0 Then Exit Function
9.      If FileDaControllare = "" Then Exit Function
```

Alla riga 6 eliminiamo gli eventuali spazi (iniziali e finali) nel nome del file, mediante la funzione *Trim*.
Alla riga 7 controlleremo l'eventuala presenza di un '*' (carattere jolly). In caso ce ne sia almeno uno verrà forzata l'uscita dalla funzione. Lo stesso tipo di controllo viene effettuato alla riga 8: in caso ci sia un '?'. Altresì, alla riga 9, controlleremo se il file da cercare non sia una stringa vuota. Anche in tal caso richiederemo l'uscita con la funzione che assumerà il valore impostato alla riga 4.

```
10.     strVerifica = Dir$(FileDaControllare, 39)
11.     If (strVerifica <> "") And (Err.Number = 0) Then FileEsiste2 = True
12. End Function
```

Alla riga 10 abbiamo la solita ricerca mediante la funzione *Dir*. Il valore 39 corrisponde alla combinazione delle costanti ***vbArchive***, ***vbHidden***, ***vbReadOnly*** e ***vbSystem*** allo scopo di ricercare anche tra i files nascosti ma non le cartelle.
Se si dovesse generare un errore nell'uso della funzione Dir$ l'esecuzione del programma continuerà in maniera silenziosa.

Pertanto alla riga successiva sarà verificato se la variabile **strVerifica** non contenga una stringa vuota e non sia stato generato qualche errore. Solo questa condizione assicura che il risultato della funzione Dir sia coerente. Al verificarsi di queste due condizioni sarà impostato il valore di uscita della funzione su True.

Questa soluzione effettua alcuni primi controlli ma ancora non provvede a riconoscere i files dalle radici dei dischi.

Terzo sistema (GetAttr)

Difficoltà: 3 / 5

Il problema principale delle funzioni viste precedentemente è che quando forniamo come parametro il percorso di una cartella contenente almeno un file, esse ci rispondono che il file esiste.

Per evitare questi errori, abbandoneremo l'uso della funzione *Dir* ed utilizzeremo al suo posto la funzione *GetAttr*. Essa ci restituisce un numero intero rappresentante gli attributi del file/cartella interessato. Per individuare lo stato di un determinato attributo è necessario

effettuare un'estrazione bit a bit del valore restituito da *GetAttr*.

```
1. Public Function FileEsiste3(ByVal FileDaControllare As String) As Boolean
2.     On Error Resume Next
3.     FileEsiste3 = False
4.
5.     FileDaControllare = Trim(FileDaControllare)
```

Anche nella funzione **FileEsiste3** eliminiamo tutti gli spazi - iniziali e finali - dal parametro *FileDaControllare*, per evitare possibili errori di scrittura. Vada da sé che la corretta apertura del file richiede il nome del file scritto in maniera corretta, ovvero senza gli spazi iniziali e finali.

```
6.     If (GetAttr(FileDaControllare) And vbDirectory) = vbDirectory Then Exit
   Function
7.     If Err.Number = 0 Then FileEsiste3 = True
8. End Function
```

Qui sta il centro di questa funzione. La funzione *GetAttr* e l'estrazione bit a bit fungono da sistema di verifica.
Alla riga 6 controlliamo che il file scelto (**FileDaControllare**) non sia una cartella. Nel caso dovesse esserlo usciremo immediatamente dalla funzione con il valore assegnato in precedenza (riga 3).
Dopo tale operazione abbiamo la completa sicurezza che il file esiste e che non si tratti di una cartella. Infatti, se il file non dovesse esistere, automaticamente la funzione *GetAttr* genererà l'errore di runtime 53 forzando l'esecuzione della riga successiva.

Poiché la riga 6 ci dà la sicurezza che il file richiesto esista, possiamo tranquillamente impostare il contenuto di **FileEsiste** a True e forzare l'uscita dalla funzione in caso non si siano verificati errori in precedenza.

Questa funzione è ancora più precisa delle precedenti, pur essendo più semplice, ma in alcuni casi - particolarmente rari - può dare qualche problema.

Quarto sistema (Open)

Difficoltà: 3 / 5

Il caso accennato sopra corrisponde ai tentativi di apertura di files che risiedono in cartelle protette, delle quali è possibile ricavare lo stato degli attributi, la presenza del file, ma non è possibile, ad esempio apportare modifiche.

La funzione che vedremo questa volta tenterà di effettuare l'apertura del file in modalità lettura e scrittura. Se l'apertura del file in tali modalità andrà a buon fine avremo la sicurezza che il file esiste e che sia accessibile, sia in lettura che in scrittura, agli altri programmi. Per qualche informazione aggiuntiva sulle modalità di apertura di un file dare uno sguardo all'HowTo dedicato alla lettura di un file.

```
1. Public Function FileEsiste4(ByVal FileDaControllare As String) As Boolean
2.     Dim FileNR As Integer
3.     On Error Resume Next
4.     FileDaControllare = Trim(FileDaControllare)
```



```
5.      FileEsiste4 = False
6.
```

Al solito, nella prima parte della funzione inizializziamo la gestione degli errori, eliminiamo gli spazi iniziali e finali dal nome del file ed inizializziamo il contenuto della funzione **FileEsiste4**.
Però in più stavolta, abbiamo dichiarato (riga 2) una variabile di nome **FileNR** che ci servirà come numero di handle del file da aprire.

```
7.      FileNR = FreeFile
8.      Open FileDaControllare For Input As FileNR
9.      Close FileNR
```

Alla riga 6 otteniamo il primo handle di file libero mediante la funzione *FreeFile* e memorizziamo tale valore all'interno della variabile **FileNR**.

Subito dopo questo tentiamo l'apertura del file in modalità *Input* ovvero lettura.
Se tale operazione andrà a buon fine avremo la completa sicurezza che il file esiste e che sia accessibile in lettura agli altri programmi. In caso contrario, l'istruzione *Open* genererà un errore di runtime che verrà verificato più in basso. Subito dopo aperto il file in lettura provvediamo a chiuderlo, per tentare la successiva riapertura in modalità scrittura.

```
10.     If Err.Number = 0 Then
11.         Open FileDaControllare For Append As FileNR
12.         Close FileNR
13.     End If
```

Ed ecco che alla riga 9 se l'operazione precedente non ha generato alcun errore, tentiamo l'apertura del file in modalità *Append*, ovvero scrittura aggiuntiva al file esistente. Giunti a questa riga, sappiamo con sicurezza che il file esiste, ed ecco perché tentiamo l'apertura in *Append*: infatti il tentativo di apertura del file in modalità *Output* comporterebbe l'azzeramento del file esistente.
È stata un po' una strada obbligatoria per preservare il contenuto del file esistente.
Anche in questo caso, se non è possibile aprire il file in tale modalità, verrà generato un errore ed il flusso del programma proseguirà regolarmente. In caso di possibile apertura il file verrà aperto.
Alla riga 11 effettuiamo, per cui, la chiusura del file aperto prima.

```
14.     If Err.Number = 0 Then FileEsiste4 = True
15. End Function
```

Giunti alla riga 11 ci basterà verificare se la precedente apertura ha generato qualche errore. Nel caso non ci siano stati errori possiamo dire con sicurezza che il file esiste ed è accessibile almeno in due modalità. Impostiamo il contenuto di **FileEsiste** a True e possiamo uscire dalla funzione.

Testiamo le quattro funzioni

A questo punto disegnamo una semplice interfaccia per provare le quattro funzioni: posizioniamo sopra un form una *TextBox* di nome **NomeFile** ed una matrice

di 4 pulsanti di nome **Controlla** con *Index* da 0 a 3.

Nella finestra del codice, alla routine che gestisce il click sopra uno qualunque dei pulsanti scriviamo:

```
 1. Private Sub Controlla_Click(Index As Integer)
 2.     Dim Esiste As Boolean
 3.     Select Case Index
 4.         Case 0: Esiste = FileEsiste1(NomeFile.Text)
 5.         Case 1: Esiste = FileEsiste2(NomeFile.Text)
 6.         Case 2: Esiste = FileEsiste3(NomeFile.Text)
 7.         Case 3: Esiste = FileEsiste4(NomeFile.Text)
 8.     End Select
 9.
10.     If Esiste = True Then
11.         MsgBox "Il file " & NomeFile.Text & " esiste", vbInformation + vbOKOnly
12.     Else
13.         MsgBox "Il file " & NomeFile.Text & " non esiste", vbCritical + vbOKOnly
14.     End If
15. End Sub
```

Il codice è molto molto semplice: alla riga 2 viene dichiarata una variabile di tipo *Boolean* di nome **Esiste**; essa conterrà il valore restituito dalle quattro funzioni di controllo del file.

Alla riga 3 viene controllato il valore di *Index* fornito come parametro dall'evento Click; tale valore corrisponde alla proprietà *Index* del pulsante premuto.
Per ognuno dei quattro pulsanti viene chiamata la funzione corrispondente viste prima.

Alla riga 10 viene controllato il valore restituito da tali funzioni: se esso è True - ovvero il file esiste, secondo il criterio adoperato dalla funzione - viene visualizzata una finestra di dialogo che informa che il file esiste. In caso contrario ne viene visualizzata una di errore.

Per provare le funzioni si consiglia l'immissione di una cartella quale **"C:\"** e vedere, per ognuna delle funzioni, il risultato riportato.

Alla fine dell'esame, la routine che può sembrare più idonea al controllo dell'esistenza di un file è l'ultima perché adopera le soluzioni di apertura sia in lettura che in scrittura per assicurare l'accesso a tale file.
Tuttavia, se in un nostro progetto non ci interessa testare la funzionalità di un file, ma soltanto elencare o controllare l'esistenza di un determinato file, quest'ultima funzione non è idonea, perché in taluni casi può portare risultati errati.
In questo genere di casi si consiglia l'utilizzo della terza funzione, ovvero il controllo mediante la funzione *GetAttr*.

Fibia FBI
11 Dicembre 2000
Rivisto e corretto il 22 Agosto 2002

Torna all'indice degli HowTo

