Manipolare file e Cartelle in VBA

VBA fornisce alcuni metodi per lavorare con i file, ma usando le funzioni di base come Dir, Name etc. che presentano una stretta correlazione di comportamento con i comandi DOS, ma purtroppo riducono notevolmente il raggio d'azione, appesantendo notevolmente il listato del codice. Per estendere le possibilità per quanto riguarda la gestione di file e directory, Microsoft ha sviluppato una serie di oggetti raggruppati all'interno della libreria **Microsoft Scripting Runtime**, e questo insieme di classe gerarchica ha una sola radice: il **FileSystemObject**, più comunemente conosciuta come **FSO**.

Il modello File System Object (FSO) è uno strumento basato sugli oggetti per l'utilizzo di cartelle e file e consente di utilizzare la sintassi *oggetto.metodo* con un numeroso gruppo di proprietà, metodi ed eventi per l'elaborazione di cartelle e file e inoltre è possibile utilizzare le istruzioni e i comandi tradizionali di Visual Basic. Il FileSystemObject è una manna per tutti gli sviluppatori che utilizzano Visual Basic, in quanto semplifica il compito di trattare con qualsiasi tipo di input e output di file e per interagire con la struttura del file System stesso.

Piuttosto che ricorrere a complesse chiamate alla API Win32, questo oggetto consente alle applicazioni di creare, modificare, spostare ed eliminare cartelle, nonché di rilevare l'esistenza ed eventualmente la posizione di cartelle specifiche, inoltre è possibile ottenere informazioni sulle cartelle, quali il nome, la data di creazione e dell'ultima modifica e così via. Il modello FSO comprende i seguenti oggetti:

FileSystemObject	Consente di creare, eliminare e gestire unità, cartelle e file e di recuperare informazioni su tali elementi.
Drive	Consente di raccogliere informazioni relative a un'unità collegata al sistema, come la quantità di spazio disponibile e il nome di condivisione. Tenere presente che "drive" nel modello FSO non corrisponde solo al termine inglese per indicare il disco rigido: può trattarsi di un'unità CD-ROM, un disco RAM e così via.
Folder	Consente di creare, eliminare o spostare cartelle e di richiedere al sistema i loro nomi, i percorsi e altre informazioni.
File	Consente di creare, eliminare o spostare file e di richiedere al sistema i loro nomi, i percorsi e altre informazioni.
TextStream	Viene utilizzato per leggere e scrivere il contenuto di un file di testo

Fig. 1

Il modello FSO è contenuto nella libreria dei vari tipi di script, che si trova nel file Scrrun.dll e se non è già presente un riferimento alla libreria, è possibile crearne uno in questo modo: dal menu **Strumenti - Riferimenti**, e nella scheda che appare scorrere la lista e selezionare la voce **Microsoft Scripting Runtime** dall'elenco, quindi fare clic su **Ok**.

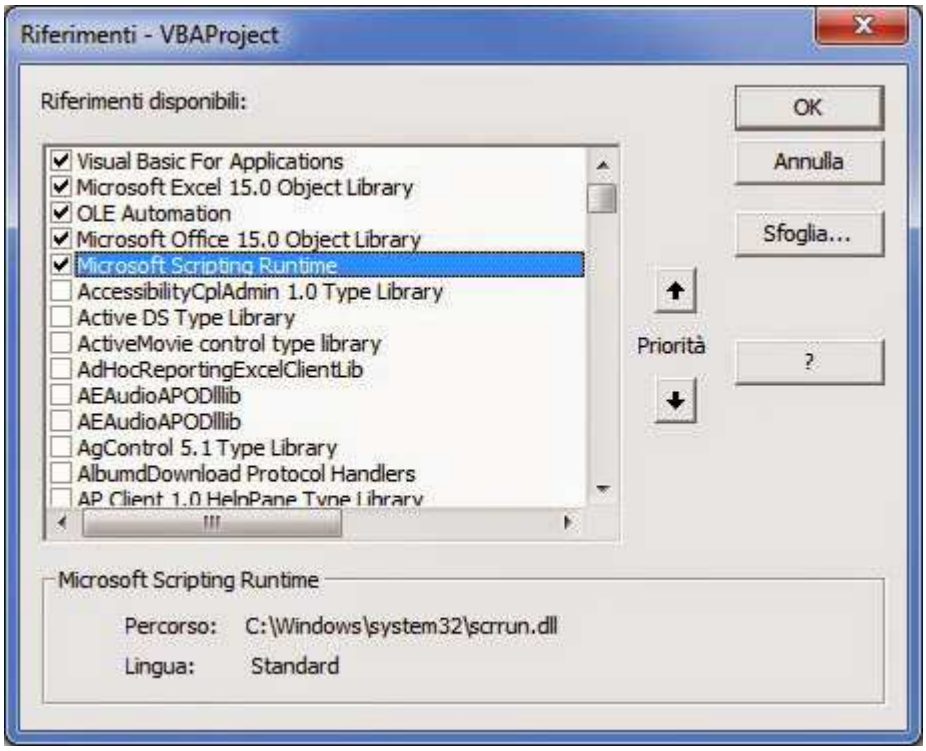

Fig. 2

E' possibile creare un oggetto FileSystemObject utilizzando il metodo CreateObject in questo modo:

```
oFSO = CreateObject("Scripting.FileSystemObject")
```

Oppure dimensionando una variabile come oggetto FileSystemObject, in questo modo:

```
Dim oFSO As New FileSystemObject
```

I metodi dell'oggetto FSO per le operazioni che può svolgere sono i seguenti:

Descrizione	Comando FSO
Creare un nuovo oggetto	**CreateFolder** o **CreateTextFile**
Eliminare un file o una cartella	**DeleteFile** o **File.Delete** oppure **DeleteFolder** o **Folder.Delete**
Copiare un oggetto	**CopyFile** o **File.Copy** oppure **CopyFolder** o **Folder.Copy**
Spostare un oggetto	**MoveFile** o **File.Move** oppure **MoveFolder** o **Folder.Move**
Accesso a un'unità, cartella o file esistente	**GetDrive**, **GetFolder** o **GetFile**

Fig. 3

Come si può notare, alcune funzioni del modello di oggetti FileSystemObject sono ridondanti, è possibile ad esempio copiare un file tramite il metodo *CopyFile* dell'oggetto FileSystemObject o tramite il metodo *Copy* dell'oggetto File. I metodi funzionano in modo identico, sono state esposte entrambe le versioni per offrire la massima flessibilità di programmazione. Non è necessario tuttavia utilizzare i metodi **Get** con i nuovi oggetti, in quanto se vengono usate le funzioni **Create** restituiscono già un puntamento a tali oggetti.

Se ad esempio viene creata una nuova cartella con il metodo CreateFolder, non è necessario utilizzare il metodo GetFolder per accedere alle sue proprietà, quali *Name*, *Path* o *Size*, è sufficiente impostare una variabile sulla funzione CreateFolder per ottenere un riferimento alla nuova cartella e quindi accedere alle sue proprietà, metodi ed eventi.

Esempio: Visualizzare le informazioni su un file utilizza alcune proprietà di FSO

```
Private Sub FileInfo(ByVal fileName As String)
    Dim fso As New FileSystemObject
    Dim fileSpec As File, mInfo As String
    Set fileSpec = fso.GetFile(fileName)
    mInfo = fileSpec.Name & vbCrLf
    mInfo = mInfo & "Creato il: "
    mInfo = mInfo & fileSpec.DateCreated & vbCrLf
    mInfo = mInfo & "Ultimo Accesso: "
    mInfo = mInfo & fileSpec.DateLastAccessed & vbCrLf
    mInfo = mInfo & "Ultima Modifica: "
    mInfo = mInfo & fileSpec.DateLastModified
    MsgBox mInfo, vbInformation, "Informazioni File"
    Set fileSpec = Nothing
End Sub
```

Che può essere richiamata con un codice come il seguente

```
Sub info1()
Dim infoF As String
infoF = "C:\Test\info1.txt"
FileInfo (infoF)
End Sub
```

Esempio: Visualizzare le informazioni su una cartella

```
Private Sub FolderInfo(ByVal folderName As String)
    Dim fso As New FileSystemObject
    Dim folderSpec As Folder, mInfo As String
    Set folderSpec = fso.GetFolder(folderName)
    mInfo = folderSpec.Name & vbCrLf
    mInfo = mInfo & "Creata il: "
    mInfo = mInfo & folderSpec.DateCreated & vbCrLf
    mInfo = mInfo & "Dimensione: "
    mInfo = mInfo & folderSpec.Size
    MsgBox mInfo, vbInformation, "Informazioni Cartella"
    Set folderSpec = Nothing
End Sub
```

Può essere richiamata in questo modo

```
Sub info2()
Dim infoC As String
infoC = "C:\Test"
FolderInfo (infoC)
End Sub
```

Esempio: Controllare se una cartella esiste

```
Function FolderE(DirName As String) As Boolean
On Error Resume Next
FolderE = GetAttr(DirName) And vbDirectory
End Function
```

La Function sopra riportata può essere testata usando un codice come il seguente

```
Sub Prova1()
Dim Percorso As String
Percorso = "C:\Test"
MsgBox FolderE(Percorso)
End Sub
```

Esempio: Controllare se un file esiste

```
Function FileE(FileName As String) As Boolean
On Error Resume Next
FileE = GetAttr(FileName) And vbArchive
End Function
```

Che può essere testata in questo modo

```
Sub Prova2()
Dim FileN As String
FileN = "C:\Test\info1.txt"
MsgBox FileE(FileN)
End Sub
```

Accedere a un disco

La collezione **Drives** dell'oggetto FileSystemObject fornisce l'accesso a tutti i record riconosciuti dal sistema operativo e ogni disco è quindi un oggetto ben distinto e il suo ID nella collezione è determinato dalla lettera di accesso (C, D, E, etc.), naturalmente, non si può scrivere, aggiungere o eliminare oggetti in questa collezione. Si deve tenere presente che usando Il metodo **DriveExists** della classe FileSystemObject è possibile determinare l'esistenza di un disco in base al suo nome.

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Set oFSO = New Scripting.FileSystemObject
If oFSO.DriveExists("C") Then
   Set oDrv = oFSO.GetDrive("C")
Else
   MsgBox "Questo disco non esiste"
End If
```

Un'altra possibilità è quella di utilizzare il riferimento diretto alla raccolta **Drives**

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Set oFSO = New Scripting.FileSystemObject
If oFSO.DriveExists("C") Then
   Set oDrv = oFSO.Drives("C")
Else
   MsgBox "Questo disco non esiste"
End If
```

Se il disco non esiste, i due metodi restituiranno lo stesso errore: *Errore # 5 chiamata di routine non valido*. Attenzione, che con questo metodo si accede ad un disco, o meglio in un disco, ma nel caso si tratti di un'unità CD-Rom e non fosse presente nessun CD non viene generato nessun errore, in sostanza non sappiamo se il dispositivo è accessibile. Esempio: Enumerare tutte le unità disco con un Loop For Each

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Set oFSO = New Scripting.FileSystemObject
For Each oDrv In oFSO.Drives
   MsgBox oDrv.DriveLetter
Next oDrv
```

Le Proprietà del disco

- *DriveLetter:* E' la lettera utilizzato dal sistema operativo per accedere al disco.
- *DriveType:* Identifica il tipo di disco, CD-Rom, Disco Fisso, Ramdisk, etc.
- *FileSystem:* E' il tipo di file System presente nel disco (es.: NTFS)
- *AvailableSpace, FreeSpace:* Indica lo spazio disponibile e lo spazio libero in byte
- *IsReady:* E' rappresentato da un valore booleano che indica se l'unità è disponibile.
- *Path:* Indica il percorso del disco.
- *RootFolder:* Corrisponde alla cartella principale, e fornisce l'accesso a tutti i file presenti sul disco.
- *SerialNumber:* Indica il numero di serie del disco.
- *ShareName:* Restituisce una stringa corrispondente alla quota del disco. Questa stringa sarà nulla se il disco non è condiviso.
- *VolumeName:* Restituisce il nome del volume (non dell'unità) in una stringa. (es.: Dati)
- *TotalSize:* dimensioni del disco in byte

Alcuni esempi di funzioni per la gestione dei dischi, iniziando dal determinare il numero di CD-ROM (compresi quello virtuale) installato sul Pc:

```
Function leggiCD1() As Integer
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Set oFSO = New Scripting.FileSystemObject
For Each oDrv In oFSO.Drives
   If oDrv.DriveType = CDRom Then leggiCD1 = leggiCD1 + 1
Next oDrv
End Function
```

Esempio: Restituire la lettera del disco rigido con più spazio disponibile

```
Function leggiD() As String
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Dim intFree As Double
Set oFSO = New Scripting.FileSystemObject
'Percorso del disco
For Each oDrv In oFSO.Drives
    'Se si tratta di un disco rigido e se contiene un filesystem valido (formattato)
    If oDrv.DriveType = Fixed And oDrv.IsReady Then
        'Se lo spazio libero è superiore a intFree, allora sostituisci
       If oDrv.FreeSpace > intFree Then
           intFree = oDrv.FreeSpace
           leggiD = oDrv.DriveLetter
        End If
    End If
Next oDrv
End Function
```

Come si può vedere in questo esempio, la manipolazione di oggetti FSO fornisce un codice strutturato nello stesso modo come se si utilizza il metodo **DAO**(Database.OpenRecordset) in cui lo stesso oggetto viene richiamato più volte. Per questo motivo, è ampiamente raccomandato **il refactoring** dei blocchi di codice in questo modo.

```
Function leggiD1() As String
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Dim intFree As Double
Set oFSO = New Scripting.FileSystemObject
'Percorso del disco
For Each oDrv In oFSO.Drives
    With oDrv
    'Se si tratta di un disco rigido e se contiene un filesystem valido (formattato)
       If .DriveType = Fixed And .IsReady Then
       'Se lo spazio libero è superiore a intFree, allora sostituisci
          If .FreeSpace > intFree Then
             intFree = .FreeSpace
             leggiD1 = .DriveLetter
          End If
       End If
End With
Next oDrv
End Function
```

Gestione Cartelle

Per accedere a una cartella si osserva lo stesso metodo usato per i dischi, in seguito vedremo i vari metodi e proprietà dell'oggetto cartella. L'accesso a un file istanziando un oggetto Folder, può essere fatto in due modi:

- Direttamente dall'oggetto FSO
- Dalla cartella principale

Prendiamo il caso di un accesso usando FSO e Il metodo **GetFolder (path)** che restituisce un oggetto *Folder*, corrispondente al percorso passato come parametro.

```
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Folder
Set oFSO = New Scripting.FileSystemObject
Set oFld = oFSO.GetFolder("C:\Windows")
```

Se però la cartella non esiste, verrà restituito un *errore 76 (percorso non trovato)*, per cui è opportuno gestirlo in questo modo.

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Folder
Set oFSO = New Scripting.FileSystemObject
' simulazione errore
Set oFld = oFSO.GetFolder("C:\Windows0")
fine:
    Exit Function
err:
    If err.Number = 76 Then
        MsgBox "Questa cartella non esiste"
    Else
        MsgBox "Errore Sconosciuto"
    End If
    Resume fine
```

Si noti che l'errore 76 può anche essere evitato controllando l'esistenza del file dal FSO con il metodo **FolderExists** in questo modo

```
Set oFSO = New Scripting.FileSystemObject
If oFSO.FolderExists("C:\Windows0") Then
    Set oFld = oFSO.GetFolder("C:\Windows0")
Else
MsgBox "Questa cartella non esiste"
End If
```

Tuttavia, non vi è alcuna garanzia che il file non venga eliminato tra il test di verifica e il tentativo di accesso, per cui è fondamentale gestire l'errore 76 con l'argomento **On Error**, oppure usando un'altra tecnica che è quella di utilizzare la gerarchia delle cartelle nel file System, in cui ogni oggetto *Folder* dispone di una proprietà **SubFolders** per consolidare le sue sottocartelle. Nel caso di *C:\Windows*, dove Windows rappresenta un "figlio" della cartella C:\ (RootFolder)

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive, oFld As Folder
Set oFSO = New Scripting.FileSystemObject
Set oDrv = oFSO.GetDrive("C")
Set oFld = oDrv.RootFolder.SubFolders("Windows")
```

Anche se a prima vista il codice sembra più complesso, o più pesante, poiché l'accesso al disco è separato dal file, con questo metodo saremo in grado di conoscere il livello di errore in caso di mancata esecuzione, in altre parole, è l'unità C, che non è disponibile o inesistente su Windows? Possiamo gestirlo in questo modo:

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive, oFld As Folder
Set oFSO = New Scripting.FileSystemObject
Set oDrv = oFSO.GetDrive("C")
Set oFld = oDrv.RootFolder.SubFolders("Windows")
fine:
    Exit Function
err:
        Select Case err.Number
            Case 5: MsgBox "Il disco non è disponibile"
            Case 76: MsgBox "Il record non esiste in questo disco"
            Case Else: MsgBox "Errore sconosciuto"
        End Select
    Resume fine
```

Come per l'accesso, è possibile utilizzare due diverse tecniche per creare una cartella:

- Utilizzando direttamente l'FSO
- Utilizzando un oggetto Folder tramite la sua collezione SubFolders

Usando FSO si può fare con questo codice:

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive, oFld As Folder
Set oFSO = New Scripting.FileSystemObject
Set oFld=oFSO.CreateFolder ("C:\Test")
fine:
    Exit Function
err:
        Select Case err.Number
            Case 58: MsgBox "Il file esiste già"
            Case 76: MsgBox "Percorso non corretto"
            Case Else: MsgBox "Errore sconosciuto"
        End Select
    Resume fine
```

Se il percorso non è valido, o la directory principale del disco è inesistente, viene generato un errore 76 (percorso non trovato), mentre se il file esiste già, l'operazione non riesce e rimanda un errore 58 (File già esistente). L'oggetto *Ofld* restituito dal metodo *CreateFolder* è riutilizzabile immediatamente dopo il codice, dal momento che la collezione *SubFolders* viene usata in questo modo:

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive
Set oFSO = New Scripting.FileSystemObject
Set oDrv = oFSO.GetDrive("C")
oDrv.RootFolder.SubFolders.Add ("Test")
fine:
   Exit Function
err:
      Select Case err.Number
         Case 5: MsgBox "Il disco non è disponibile"
         Case 58: MsgBox "Il file esiste già"
         Case 76: MsgBox "Percorso non corretto"
         Case Else: MsgBox "Errore sconosciuto"
      End Select
   Resume fine
```

Come si può vedere dal codice, la differenza sta solo nella gestione degli errori. Per chi ha familiarità con DAO, probabilmente si ricorderà la proprietà Attributes per oggetti diversi che rappresenta l'aggiunta logica dei diversi valori degli elementi di qualificazione. Ad esempio, una cartella potrebbe essere nascosta, oppure nascosta e archiviata, etc. I valori possibili sono:

- *Normal*
- *ReadOnly*: Sola lettura
- *Hidden*: Cartella nascosta
- *System*: Cartella Sistema
- *Archive*: archivio di file
- *Compressed*: Cartella compressa

Di seguito sono riportati alcuni possibili listati di prova:

```
If oFld.Attributes And Directory Then MsgBox "Nascosto"
If oFld.Attributes And (Hidden + ReadOnly) Then MsgBox "Nascosta e di sola lettura"
```

La proprietà Attributes può essere applicata in lettura o in scrittura, il che significa che è possibile modificare gli attributi di cartella. Esempio per rimuovere una modalità cartella nascosta:

```
If oFld.Attributes And Hidden Then
 oFld.Attributes = oFld.Attributes - Hidden
```

Le proprietà dell'oggetto Folder

- *Attributes*: Come visto sopra, attribuisce la cartella.
- *DateCreated*: Data di creazione della cartella
- *DateLastAccessed*: Data dell'ultimo accesso alla cartella
- *DateLastModified*: Data ultima modifica
- *Drive*: Corrispondente al disco in cui si trova la cartella
- *Files*: E' una raccolta di file nella cartella
- *IsRootFolder*: E' un valore booleano che determina se la cartella è la radice del disco
- *Name*: Indica il nome della cartella
- *ParentFolder*: Folder corrisponde alla cartella principale, se la cartella è una cartella RootFolder questa proprietà restituisce Nothing.
- *Path*: Indica il percorso completo della cartella
- *ShortName*: E' il nome "breve" di una cartella con un massimo di 8 caratteri
- *ShortPath*: percorso completo della cartella in cui ogni componente è conforme alla norma ShortName
- *Size*: Dimensione totale del file in byte. Questa è la somma delle dimensioni di tutti i file nella cartella e relative sottocartelle.
- *SubFolders*: Indica un raggruppamento di sottocartelle
- *Type*: Tipo di file. In tutti i casi esaminati, è FileFolder

Il metodo Copy dell'oggetto Folder

Il metodo Copy consente di copiare la cartella e il suo contenuto in un altro percorso (esistenti o meno). Sintassi: *Copy(Destination As String, [OverWriteFiles As Boolean = True])*. Dove *Destination* rappresenta un percorso valido di destinazione della copia e se *OverWriteFiles* è vero, i file presenti nella directory di destinazione vengono sovrascritti se hanno lo stesso nome. Esempio: *oFld.Copy "C:\Test", True*

Se il percorso di destinazione non è corretto, viene generato un errore 76 (percorso non trovato) e se OverWriteFiles è False e la destinazione contiene già dei file con lo stesso nome, viene generato un errore 58 (File già esistente). Il metodo CopyFolder dell'oggetto FSO riproduce lo stesso comportamento. Esempio: *oFSO.CopyFolder("C:\Test","C:\Test2",True)*

Il metodo Delete dell'oggetto Folder

Il metodo Delete rimuove la cartella specificata. Sintassi: *Delete([Force As Boolean = False])*. Il parametro Force se viene collocato a True esegue una forzatura per eliminare file di sola lettura nella cartella specificata e relative sottocartelle. Se invece Force è False, viene generato un errore 70 (Permesso negato) e il file diventa di sola lettura, e vale anche se il file è aperto. Esempio: *oFld.Delete False*

E 'possibile utilizzare un altro metodo per eliminare una cartella usando il metodo DeleteFolder dell'oggetto FileSystemObject. Esempio: *oFSO.DeleteFolder "C:\Test", False*

Il metodo Move dell'oggetto Folder

Il metodo Move sposta la cartella di destinazione specificata in un altro percorso.
Sintassi: *Move(Destination As String)*. Esempio: *oFld.Move "C:\Test2"*

E 'possibile spostare una cartella in un'altra, tuttavia, inoltre se il contenuto esiste già nella destinazione verrà generato un errore 58. Anche in questo caso, si può utilizzare l'equivalente FSO con il metodo MoveFolder. Esempio: *oFSO.MoveFolder "C:\Test", "C:\Testi2"*

Cartelle speciali

Il metodo *GetSpecialFolder* consente l'accesso a directory specifiche.
Sintassi: *GetSpecialFolder(SpecialFolder As SpecialFolderConst) As Folder* e i valori possibili per SpecialFolder sono:

- *WindowsFolder*: Cartella in cui è installato Windows
- *SystemFolder*: Cartella di Sistema (Windows)
- *TemporaryFolder*: Cartella per memorizzare i file temporanei

Gestione dei file

Questa è la gerarchia più bassa dell'elemento e ogni file è rappresentato da un oggetto file nell'insieme Files in un oggetto Folder. Per accedere a un file si possono essere utilizzati due metodi per restituire un oggetto File.

- Utilizzando il metodo GetFile del FileSystemObject
- Utilizzando la collezione File di un oggetto Folder

Usando FSO può essere espresso in questo modo

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Set oFSO = New Scripting.FileSystemObject
Set oFl = oFSO.GetFile("C:\Test\info1.txt")
fine:
    Exit Function
err:
        Select Case err.Number
            Case 53: MsgBox "Il file non è stato trovato"
            Case Else: MsgBox "Errore sconosciuto"
        End Select
    Resume fine
```

Si deve prestare attenzione che se il percorso del file non è corretto, viene generato l'errore 53 (File non trovato). In questi casi viene usato il *metodoFileExists* per verificare l'esistenza del file.

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Set oFSO = New Scripting.FileSystemObject
If oFSO.FileExists("C:\Test\info1.txt") Then
    Set oFl = oFSO.GetFile("C:\Test\info1.txt")
End If
fine:
    Exit Function
err:
        Select Case err.Number
            Case 53: MsgBox "Il file non è stato trovato"
            Case Else: MsgBox "Errore Sconosciuto"
        End Select
    Resume fine
```

Mentre usando l'oggetto Folder il codice è il seguente:

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Scripting.Folder
Dim oFl As Scripting.File
Set oFSO = New Scripting.FileSystemObject
Set oFld = oFSO.GetFolder("C:\Test")
Set oFl = oFld.Files("info1.txt")
fine:
   Exit Function
err:
      Select Case err.Number
         Case 76: MsgBox "La cartella non esiste"
         Case 53: MsgBox "Il file non si trova in questa cartella"
         Case Else: MsgBox "Errore Sconosciuto"
      End Select
   Resume fine
```

Le proprietà dell'oggetto File

- *Attributes*: Gli attributi dei file, stessa proprietà per le cartelle.
- *DateCreated*: Data di creazione del file
- *DateLastAccessed*: Data dell'ultimo accesso
- *DateLastModified*: Data ultima modifica
- *Drive*: Indica l'unità corrispondente al disco in cui risiede il file
- *Name*: Nome del file.
- *ParentFolder*: E' la Cartella che contiene il file.
- *Path*: Indica il percorso completo del file.
- *ShortName*: Denominazione rispettando lo standard 8.3 (8 caratteri per il nome e 3 per l'estensione).
- *ShortPath*: Percorso completo della cartella in cui ogni componente è conforme alla norma ShortName
- *Size*: Dimensione in byte del file
- *Type*: Tipo di file

Il metodo Copy dell'oggetto File

Il metodo copy permette di copiare il file in un'altra destinazione. Sintassi: *Copy(Destination As String, [OverWriteFiles As Boolean = True])*

Dove *Destination* è un percorso valido del file copiato, e se esiste già un file con lo stesso nome, verrà sovrascritto con l'unica condizione che OverWriteFiles sia uguale a true. Esempio: *oFl.Copy "C:\Test2\info1.txt", True*

Se il percorso di destinazione non è corretto, viene generato un *errore 76* (percorso non trovato). Se invece *OverWriteFiles* è uguale a False e il file esiste già, viene generato un *errore 58* (File già esistente). Il metodo CopyFile dell'oggetto FSO riproduce lo stesso comportamento. Esempio: *oFSO.CopyFile("C:\Test\info1.txt","C:\Test2\info1. txt ",True)*

Il metodo Delete dell'oggetto File

Il metodo Delete elimina il file specificato. Sintassi: *Delete([Force As Boolean = False])*, dove il parametro *Force* se posto uguale a True, forza l'eliminazione di un file anche se è di sola lettura, se invece è uguale a False e il file è di sola lettura, viene generato un *errore 70* (Autorizzazione negata). Questo errore viene generato se il file è aperto. Esempio: *oFl.Delete False*. È anche possibile utilizzare il metodo DeleteFile per l'oggetto FSO. Esempio: *oFSO.DeleteFile "C:\Test\info1.txt", False*

Il metodo Move dell'oggetto File

Il metodo Move permette di spostare il file in un'altra destinazione. Sintassi: *Move(Destination As String)*. Esempio: *oFl.Move "C:\Test2\info1.txt"*. In alternativa, si può usare il metodo MoveFile dell'oggetto FileSystemObject. Esempio: *oFSO.MoveFile "C:\Test\info1.txt ", "C:\Test2\info1.txt "*

Questi metodi possono essere usati anche per rinominare un file.
Esempio: *oFSO.MoveFile "C:\Test\info1.txt ", "C:\Test\info1_old.txt "*

Verificare l'esistenza di un percorso

Sebbene il FileSystemObject fornisce diversi metodi per verificare l'esistenza di un file, è difficile in caso di errore sapere quale parte del percorso ha causato il problema in caso di errore. Immaginate questo percorso: *C:\Test\Test2\Prove3* dove Prove3 non esiste. Usando il metodo FolderExists viene restituito il valore di False, ma l'utente ignorerà se è stato causato da Test, Test2 o Prove3. In questo caso si può usare una funzione più completa per identificare il file alla radice dell'errore.

```
Function TestC(strC As String) As Boolean
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject, oFld As Scripting.Folder
Dim oDrv As Scripting.Drive, i As Integer
Dim strD() As String
'istanziare FSO
Set oFSO = New Scripting.FileSystemObject
'Accedere al disco
Set oDrv = oFSO.Drives(oFSO.GetDriveName(strC))
'Instanziare la cartella principale
Set oFld = oDrv.RootFolder
'tagliare il percorso della cartella
strD = Split(strC, "\")
'tentativi di accedere alle sotto cartelle
For i = 1 To UBound(strD) - 1
   Set oFld = oFld.SubFolders(strD(i))
Next i
TestC = True
fine:
   Exit Function
err:
   Select Case err.Number
      Case 5: MsgBox "Il disco non esiste"
      Case 76: MsgBox "Impossibile trovare il file: " & strD(i)
      Case Else: MsgBox "Errore Sconosciuto"
   End Select
   Resume fine
End Function
```