

Manipulation des fichiers en VBA

par [Christophe WARIN](#)

Date de publication : 7/12/2006

Dernière mise à jour : 7/12/2006


A travers ce document, je vous propose d'étudier les différentes techniques de manipulation des fichiers en VBA. Au sommaire : Gestion des disques, des dossiers, des fichiers Manipulation des fichiers textes Recherche de fichiers

- I - Le FileSystemObject
 - I-A - Introduction
 - I-B - Gestion des disques
 - I-B-1 - Accéder à un disque
 - I-B-2 - Lister les disques
 - I-B-3 - Les propriétés des disques
 - I-B-4 - Quelques exemples de traitements sur les disques
 - I-C - Gestion des dossiers
 - I-C-1 - Accéder à un dossier
 - I-C-2 - Créer un dossier
 - I-C-3 - Les attributs des dossiers
 - I-C-4 - Les propriétés de l'objet Folder
 - I-C-5 - Les méthodes de l'objet Folder
 - I-C-5-a - Copy
 - I-C-5-b - Delete
 - I-C-5-c - Move
 - I-C-5-d - CreateTextFile
 - I-C-6 - Les dossiers spéciaux
 - I-D - La gestion des fichiers
 - I-D-1 - Accéder à un fichier
 - I-D-2 - Les propriétés de l'objet File
 - I-D-3 - Les méthodes de l'objet File
 - I-D-3-a - Copy
 - I-D-3-b - Delete
 - I-D-3-c - Move
 - I-D-3-d - OpenAsTextStream
 - I-E - Les méthodes du FSO
 - I-E-1 - BuildPath
 - I-E-2 - GetDriveName, GetFileName, GetBaseName, GetExtensionName
 - I-E-3 - GetParentFolder
 - I-E-4 - GetTempName
 - I-E-5 - Tester l'existence d'un chemin
- II - Les fichiers textes
 - II-A - Introduction
 - II-B - Accès séquentiel
 - II-B-1 - Lecture
 - II-B-2 - Ecriture
 - II-C - Accès direct
 - II-D - Les TextStream
 - II-D-1 - Ouvrir un fichier texte
 - II-D-2 - Lecture
 - II-D-3 - Ecriture
 - II-D-4 - Fermeture
- III - La recherche de fichiers
 - III-A - Recherche recursive en utilisant le FSO
 - III-B - L'objet FileSearch
 - III-B-1 - Recherche basique
 - III-B-2 - Recherche complexe
- IV - Conclusion

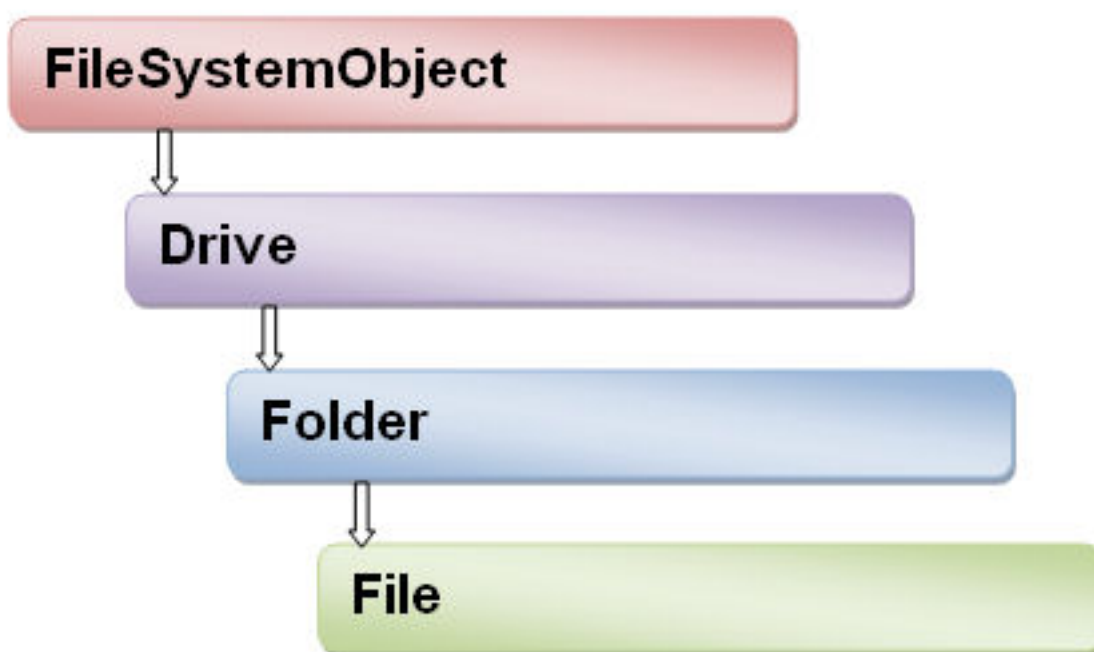
I - Le FileSystemObject

I-A - Introduction

VBA propose des méthodes pour travailler sur les fichiers. Ces fonctions de bases telles que Dir, Name, etc. ont un comportement proche des commandes DOS. Malheureusement, elles sont beaucoup trop limitées et permettent, hélas, que très peu d'actions ou bien alors au prix d'un code lourd à maintenir. Afin d'étendre les possibilités en ce qui concerne la manipulation des fichiers et des répertoires, Microsoft a mis au point un ensemble d'objets regroupés au sein de la librairie **Microsoft Scripting Runtime**. Cet ensemble de classe hiérarchique possède une unique racine : le **FileSystemObject**, plus communément connu sous le nom de **FSO**.

 Bien qu'il s'agisse en fait du nom d'une classe, il n'est pas rare de voir le mot **FSO** désigner la technique d'accès aux fichiers dans sa globalité.

La hiérarchie des objets de cette librairie peut être comparée à celle de l'explorateur Windows : des fichiers inclus dans des dossiers qui eux-mêmes sont inclus dans des disques. Le **FSO** peut alors être illustré par le poste de travail : il donne accès aux disques.



Etant donné que le **FSO** est la base de tout ce modèle, il ne peut dériver d'aucun objet. Par conséquent, il doit être instancié par le mot clé **New**. Les autres objets seront instanciés par des fonctions de cet élément maître (ou d'autres sous-éléments).

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation d'un objet enfant
Set oDrv = oFSO.Drives(1)
```

N'oubliez pas d'ajouter la référence **Microsoft Scripting Runtime** à votre projet sans quoi une erreur sera levée.

I-B - Gestion des disques

La collection **Drives** de l'objet **FileSystemObject** donne accès à l'ensemble des disques reconnus par le système d'exploitation. Chaque disque correspond alors un objet **Drive** distinct et son identifiant au sein de la collection est déterminé par sa lettre d'accès (C, D, E, etc.).

Bien entendu, vous ne pouvez pas agir en écriture sur cette collection. L'ajout et la suppression d'objet y est donc impossible.

I-B-1 - Accéder à un disque

La méthode **DriveExists** exposée par la classe **FileSystemObject** teste l'existence d'un disque en fonction de son nom.

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation d'un objet drive correspondant au disque C
If oFSO.DriveExists("C") Then
    Set oDrv = oFSO.GetDrive("C")
Else
    MsgBox "Ce disque n'existe pas"
End If
```

Une autre possibilité est d'utiliser directement la référence à la collection **Drives** :

```
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation d'un objet drive correspondant au disque C
If oFSO.DriveExists("C") Then
    Set oDrv = oFSO.Drives("C")
Else
    MsgBox "Ce disque n'existe pas"
End If
```

Si le disque n'existe pas, les deux méthodes lèveront la même erreur : erreur n°5, appel de procédure incorrect.

Attention, il s'agit d'accéder ici à un disque ou plutôt à une unité de disque. Si aucun CD n'est présent dans le lecteur de CD-ROM, aucune erreur ne sera générée. A ce stade, on ne sait pas si l'unité est accessible.

I-B-2 - Lister les disques

Une boucle **For Each** permet d'énumérer l'ensemble des unités de disques :

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

I-B-3 - Les propriétés des disques

DriveLetter : Lettre utilisé par le système d'exploitation pour accéder au disque. Il s'agit de la clé de la collection Drives. Exemple : D

DriveType : Type du disque (CDRom,Fixed,RamDisk,Remote,Removable,UnknownType)

FileSystem : Type du système de fichier du disque. Exemple : NTFS.

AvailableSpace, FreeSpace : Espace disponible et espace libre en octets

IsReady : Booléen indiquant si l'unité de disque est disponible. Dans le cas d'un lecteur de CD-ROM, elle permettra de savoir si un disque est présent ou pas.

Path : Chemin de l'unité de disque. Exemple : "D :"


RootFolder : Retourne un objet Folder correspondant au dossier racine du disque. C'est cet objet qui donne accès à l'ensemble des autres dossiers présent sur le disque. Le path de ce dossier est alors : Drive.Path & \. Exemple D:\

SerialNumber : Numéro de série du disque. Exemple : 1154367849

ShareName : Retourne une chaîne de caractères correspondant au nom de partage du disque. Cette chaîne sera nulle si le disque n'est pas partagé. Exemple : Partage_Donnees

VolumeName : Retourne le nom du volume (et non de l'unité) dans une chaîne de caractères. Exemple : Donnees

TotalSize : Taille du disque en octets

 *Les objets **Drive** ne possède pas de méthodes.*

I-B-4 - Quelques exemples de traitements sur les disques

Voici un exemple de fonction permettant de connaître le nombre de lecteur de CD-ROM (virtuels compris) installés sur la machine :

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

Autre exemple : Retourner la lettre du disque dur contenant le plus de place disponible. Cela peut s'avérer pratique pour une sauvegarde.

```
Function LecteurLibre() As String
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Scripting.Drive
Dim intFree As Double
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Parcours les disques
```

```

For Each oDrv In oFSO.Drives
    'Si c'est un disque dur et s'il contient un filesystem valide (formaté)
    If oDrv.DriveType = Fixed And oDrv.IsReady Then
        'Si la taille libre est supérieure à intFree, alors remplacer
        If oDrv.FreeSpace > intFree Then
            intFree = oDrv.FreeSpace
            LecteurLibre = oDrv.DriveLetter
        End If
    End If
Next oDrv
End Function

```

Comme vous pouvez le constater dans cet exemple, la manipulation des objets **FSO** donne un code structuré de la même façon qu'avec **DAO** : le même objet est appelé de nombreuses fois. Pour cette raison, il est largement conseillé de factoriser le code avec des blocs **With**.

```

Function LecteurLibre() As String
    Dim oFSO As Scripting.FileSystemObject
    Dim oDrv As Scripting.Drive
    Dim intFree As Double
    'Instanciation du FSO
    Set oFSO = New Scripting.FileSystemObject
    'Parcours les disques
    For Each oDrv In oFSO.Drives
        With oDrv
            'Si c'est un disque dur et s'il contient un filesystem valide (formaté)
            If .DriveType = Fixed And .IsReady Then
                'Si la taille libre est supérieure à intFree, alors remplacer
                If .FreeSpace > intFree Then
                    intFree = .FreeSpace
                    LecteurLibre = .DriveLetter
                End If
            End If
        End With
    Next oDrv
End Function

```

I-C - Gestion des dossiers

Comme pour les disques, nous allons respecter le même plan. Dans un premier temps nous allons voir comment accéder à un dossier et nous traiterons ensuite de l'ensemble des méthodes et propriétés des objets **Folder**.

I-C-1 - Accéder à un dossier

L'accès à un dossier (entendez par là instancier un objet **Folder**), peut se faire de deux manières :

- Directement depuis l'objet **FSO**
- Via un dossier parent

Prenons le cas de l'accès par **FSO**. La méthode **GetFolder(Chemin)** retourne un objet **Folder** correspondant au chemin passé en paramètre.

```

Dim oFSO As Scripting.FileSystemObject
Dim oFld As Folder

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Accède au dossier
Set oFld = oFSO.GetFolder("D:\Windows")

```

Si le dossier n'existe pas, une erreur 76 (Chemin d'accès introuvable) sera levée. Elle doit être traitée.

```

On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Folder

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Accède au dossier
Set oFld = oFSO.GetFolder("D:\Windows0")

fin:

    Exit Function

err:
'Gestion de l'erreur 76
If err.Number = 76 Then
    MsgBox "Ce dossier n'existe pas"
Else
    MsgBox "Erreur inconnue"
End If

Resume fin

```

A noter que l'erreur 76 peut aussi être évitée en vérifiant l'existence du dossier depuis le **FSO**. (Méthode **FolderExists**)

```

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Accède au dossier
If oFSO.FolderExists("D:\Windows0") Then
    Set oFld = oFSO.GetFolder("D:\Windows0")
Else
    MsgBox "Ce dossier n'existe pas"
End If

```

Cependant, rien ne garantit que le dossier ne soit pas supprimé entre le test et la tentative d'accès. Selon moi, la gestion de l'erreur 76 à l'aide d'**On Error** est donc impérative.

Comme indiqué au début de ce chapitre, l'autre technique consiste à utiliser la hiérarchie des dossiers au sein du système de fichiers. Chaque objet **Folder** possède une propriété **SubFolders** regroupant ses sous-dossiers. Dans le cas de *D:\Windows*, *Windows* est un dossier enfant du dossier *D:* (le **RootFolder** de l'objet **Drive D**)

```

Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive
Dim oFld As Folder

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation du disque
Set oDrv = oFSO.GetDrive("D")
'Accède au dossier Windows de D
Set oFld = oDrv.RootFolder.SubFolders("Windows")

```

Certes, à première vue cela semble plus complexe (ou tout du moins plus lourd). Cependant, du fait que l'accès au disque est distinct de celui du dossier, nous serons capables de connaître le niveau d'erreur en cas d'échec. En d'autres termes, est-ce le disque D qui est indisponible ou bien le répertoire Windows inexistant ?

```

On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive
Dim oFld As Folder

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation du disque
Set oDrv = oFSO.GetDrive("D")
'Accède au dossier Windows de D
Set oFld = oDrv.RootFolder.SubFolders("Windows")

```

```

fin:
    Exit Function

err:
    Select Case err.Number
        Case 5: MsgBox "Le disque n'est pas disponible"
        Case 76: MsgBox "Le dossier n'existe pas dans ce disque"
        Case Else: MsgBox "Erreur inconnue"
    End Select

Resume fin

```

I-C-2 - Créer un dossier

Comme pour l'accès, il est possible d'utiliser deux techniques différentes pour créer un dossier :

- Depuis le **FSO** directement
- Depuis un autre objet **Folder** (via sa collection **SubFolders**)

Depuis le **FSO** :

```

On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive
Dim oFld As Folder

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Crée le repertoire
Set oFld=oFSO.CreateFolder ("D:\Essai")

fin:
    Exit Function

err:
    Select Case err.Number
        Case 58: MsgBox "Le dossier existe déjà"
        Case 76: MsgBox "Chemin incorrect"
        Case Else: MsgBox "Erreur inconnue"
    End Select

Resume fin

```

Si le chemin est incorrect (répertoire parent inexistant, disque inexistant), une erreur 76 (Chemin introuvable) est levée. Si le dossier existe déjà, l'opération échoue avec l'erreur 58 (Le fichier existe déjà). L'objet **oFld** retourné par la méthode **CreateFolder** est réutilisable immédiatement dans la suite du code.

Depuis la collection **SubFolders** :

```

On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oDrv As Drive

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject

'Instanciation du disque
Set oDrv = oFSO.GetDrive("D")
'Crée le repertoire
oDrv.RootFolder.SubFolders.Add ("Essai")

fin:
    Exit Function

```

```
err:
    Select Case err.Number
        Case 5: MsgBox "Le disque n'est pas disponible"
        Case 58: MsgBox "Le dossier existe déjà"
        Case 76: MsgBox "Chemin incorrect"
        Case Else: MsgBox "Erreur inconnue"
    End Select

Resume fin
```

La différence réside une nouvelle fois dans la gestion d'erreur.

I-C-3 - Les attributs des dossiers

Si vous êtes familier avec **DAO**, vous vous rappelez sans doute de la propriété **Attributes** des différents objets. Pour rappel, il s'agit de l'addition logique de plusieurs valeurs qualifiant un objet.

Par exemple, un dossier peut être caché, caché ET archivé, etc.

Les valeurs possibles sont :

- **Normal**
- **ReadOnly** : Dossier en lecture seule
- **Hidden** : Dossier caché
- **System** : Dossier système
- **Archive** : Dossier archivé
- **Compressed** : Dossier compressé

Ci-dessous quelques tests possibles :

```
'Teste si le dossier est caché
If oFld.Attributes And Directory Then MsgBox "Caché"
'Teste si me dossier est caché et en lecture seule
If oFld.Attributes And (Hidden + ReadOnly) Then MsgBox "Caché et en lecture seule"
```

La propriété **Attributes** est en lecture/écriture, cela signifie que vous pouvez modifier les attributs du dossier.

Exemple pour retirer le mode caché d'un dossier :

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

I-C-4 - Les propriétés de l'objet Folder

Attributes : Comme vu précédemment, attributs du dossier.

DateCreated : Date de création du dossier

DateLastAccessed : Date du dernier accès au dossier

DateLastModified : Date de dernière modification

Drive : Objet **Drive** correspondant à l'unité de disque d'où est issu le dossier.

Files : Collection regroupant les fichiers du dossier

IsRootFolder : Booléen qui définit si le dossier est le dossier racine de son unité de disque.

Name : Nom du dossier. Exemple : Windows

ParentFolder : Objet **Folder** correspondant au dossier parent. Si le dossier est un dossier **RootFolder** cette propriété retourne **Nothing**.

Path : Chemin complet d'accès au dossier. Exemple : D:\Windows

ShortName : Nom court sur 8 caractères maximum. Exemple ESSAI1~1

ShortPath : Chemin complet d'accès au dossier où chaque composant respecte la norme évoquée pour **ShortName**. Exemple D:\ABCDEF1~1\ESSAI1~1

Size : Taille totale du dossier en octets. Il s'agit de la somme de la taille de tous les fichiers présents dans le dossier et ses sous-dossiers.

SubFolders : Collection d'objet **Folder** regroupant les sous-dossiers.

Type : Type du dossier. Dans tous les cas testés, il s'agit de **FileFolder**.

I-C-5 - Les méthodes de l'objet Folder

I-C-5-a - Copy

La méthode **Copy** copie le dossier et son contenu dans un autre (existant ou non).

Syntaxe :

```
Copy(Destination As String, [OverWriteFiles As Boolean = Vrai])
```

Destination correspond au chemin valide de destination de la copie. Si **OverWriteFiles** est à **True**, les fichiers déjà présents dans le répertoire de destination sont écrasés s'ils portent le même nom.

Exemple :

```
oFld.Copy "D:\Essai2", True
```

Si le chemin de destination est incorrect, une erreur 76 (Chemin introuvable) est levée. Si **OverWriteFiles** est à **False** et que la destination possède déjà des fichiers du même nom, une erreur 58 (Le fichier existe déjà) est générée.

La méthode **CopyFolder** de l'objet **FSO** reproduit le même comportement.

Exemple :

```
oFSO.CopyFolder("D:\Essai", "D:\Essai2", True)
```

I-C-5-b - Delete

La méthode **Delete** supprime le dossier spécifié.

Syntaxe :

```
Delete([Force As Boolean = Faux])
```

Le paramètre **Force** placé à **True** force la suppression des fichiers en lecture seule présents dans le dossier et ses sous-dossiers. Si **Force** est à **False** et que certains fichiers sont en lecture seule, une erreur 70 (Permission refusée) est générée. Il en va de même si un fichier est ouvert.

Exemple :

```
oFld.Delete False
```

Il est possible d'utiliser une autre méthode pour supprimer un dossier : la méthode **DeleteFolder** de l'objet **FileSystemObject**.

Exemple :

```
oFSO.DeleteFolder "D:\Essai", False
```

I-C-5-c - Move

La méthode **Move** déplace le dossier spécifié vers une autre destination.

Syntaxe :

```
Move(Destination As String)
```

Comme pour la méthode **Delete**, il ne faut pas qu'un fichier soit ouvert. Si c'est le cas l'erreur 70 sera levée.

Exemple :

```
oFld.Move "D:\essai2"
```

Il est possible de déplacer un dossier vers un autre. Toutefois, il ne faut pas que son contenu existe déjà dans la cible sans quoi une erreur 58 sera levée.

Là encore, il existe un équivalent utilisable depuis l'objet **FSO** avec la méthode **MoveFolder**

Exemple :

```
oFSO.MoveFolder "D:\Essai", "D:\Essai2"
```

I-C-5-d - CreateTextFile

La méthode **CreateTextFile** permet de créer un fichier texte. Elle sera étudiée en détails plus loin dans ce tutoriel.

I-C-6 - Les dossiers spéciaux

La méthode **GetSpecialFolder** permet d'accéder à des répertoires particuliers.

Syntaxe :

```
GetSpecialFolder(SpecialFolder As SpecialFolderConst) As Folder
```

Les valeurs possibles de **SpecialFolder** sont :

- **WindowsFolder** : Dossier où Windows est installé
- **SystemFolder** : Dossier System (dans Windows)
- **TemporaryFolder** : Dossier où stocker les fichiers temporaires

I-D - La gestion des fichiers

Il s'agit là de l'élément le plus bas de la hiérarchie. Chaque fichier est représenté par un objet **File** au sein de la collection **Files** d'un objet **Folder**.

I-D-1 - Accéder à un fichier

Une nouvelle fois, deux méthodes peuvent être utilisées pour retourner un objet **File**.

- A l'aide de la méthode **GetFile** du **FileSystemObject**
- Depuis la collection **Files** d'un objet **Folder**

Depuis **FSO** :

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation de l'objet File
Set oFl = oFSO.GetFile("D:\Essai\MonFichier.txt")

fin:
Exit Function

err:
Select Case err.Number
Case 53: MsgBox "Le fichier est introuvable"
Case Else: MsgBox "Erreur inconnue"
End Select

Resume fin
```

Attention, si le chemin d'accès au fichier est incorrect, l'erreur 53 (Fichier introuvable) est levée.

La méthode **FileExists** permet de tester l'existence du fichier.

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation de l'objet File
If oFSO.FileExists("D:\Essai\MonFichier.txt") Then
    Set oFl = oFSO.GetFile("D:\Essai\MonFichier.txt")
End If

fin:
Exit Function

err:
    Select Case err.Number
        Case 53: MsgBox "Le fichier est introuvable"
        Case Else: MsgBox "Erreur inconnue"
    End Select

Resume fin
```

Depuis un dossier :

```
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Scripting.Folder
Dim oFl As Scripting.File

'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
'Instanciation du dossier
Set oFld = oFSO.GetFolder("D:\Essai")
'Instanciation de l'objet File
Set oFl = oFld.Files("MonFichier.txt")

fin:
Exit Function

err:
    Select Case err.Number
        Case 76: MsgBox "Le dossier n'existe pas"
        Case 53: MsgBox "Le fichier est introuvable dans ce dossier"
        Case Else: MsgBox "Erreur inconnue"
    End Select

Resume fin
```

I-D-2 - Les propriétés de l'objet File

Attributes : Attributs du fichier. Identique à la propriété **Attributes** des dossiers.

DateCreated : Date de création du fichier

DateLastAccessed : Date de dernier accès au fichier

DateLastModified : Date de dernière modification

Drive : Objet **Drive** correspondant à l'unité de disque d'où est issu le fichier.

Name : Nom du fichier. Exemple : monfichier.txt

ParentFolder : Objet **Folder** correspondant au dossier contenant le fichier.

Path : Chemin complet d'accès au fichier. Exemple : D:\Windows\monfichier.txt

ShortName : Nom court respectant la norme 8.3 (8 caractères pour le nom, 3 pour l'extension). Exemple : MONFIC~1.TXT

ShortPath : Chemin complet d'accès au dossier où chaque composant respecte la norme évoquée pour **ShortName**. Exemple D:\ABCDEF1~1\ MONFIC~1.TXT

Size : Taille en octet du fichier

Type : Type du fichier. Exemple : Document Texte. C'est ce type qui est affiché dans l'explorateur Windows.

I-D-3 - Les méthodes de l'objet File

I-D-3-a - Copy

La méthode **Copy** copie le fichier vers une autre destination

Syntaxe :

```
Copy(Destination As String, [OverWriteFiles As Boolean = Vrai])
```

Destination correspond au chemin valide de destination de la copie. Si un fichier portant le même nom existe déjà, il sera écrasé à l'unique condition qu'**OverWriteFiles** soit égal à **True**.

Exemple :

```
oF1.Copy "D:\Essai2\monfichier.txt", True
```

Si le chemin de destination est incorrect, une erreur 76 (Chemin introuvable) est levée. Si **OverWriteFiles** est à **False** et que le fichier est déjà présent, une erreur 58 (Le fichier existe déjà) est générée.

La méthode **CopyFile** de l'objet **FSo** reproduit le même comportement.

Exemple :

```
oFSo.CopyFile("D:\Essai\monfichier.txt", "D:\Essai2\ monfichier.txt ", True)
```

I-D-3-b - Delete

La méthode **Delete** supprime le fichier.

Syntaxe :

```
Delete([Force As Boolean = Faux])
```

Le paramètre **Force** placé à **True** force la suppression du fichier même s'il est en lecture seule. S'il est égal à **False** et que le fichier est en lecture seule, une erreur 70 (Permission refusée) est générée. Cette même erreur sera levée si le fichier est ouvert.

Exemple :

```
oFl.Delete False
```

Il est aussi possible d'utiliser la méthode **DeleteFile** du **FSO**.

Exemple :

```
oFSO.DeleteFile "D:\Essai\monfichier.txt", False
```

I-D-3-c - Move

La méthode **Move** déplace le fichier vers une autre destination.

Syntaxe :

```
Move(Destination As String)
```

Exemple :

```
oFl.Move "D:\essai2\monfichier.txt"
```

Autre possibilité : la méthode **MoveFile** du **FileSystemObject**

Exemple :

```
oFSO.MoveFile "D:\Essai\ monfichier.txt ", "D:\Essai2\ monfichier.txt "
```

Ces méthodes peuvent être utilisées pour renommer un fichier.

Exemple :

```
oFSO.MoveFile "D:\Essai\ monfichier.txt ", "D:\Essai\ Old_monfichier.txt "
```

I-D-3-d - OpenAsTextStream

Cette méthode permet d'ouvrir un fichier texte. Elle sera décrite plus loin dans ce document.

I-E - Les méthodes du FSO

L'objet **FileSystemObject** offre quelques méthodes intéressantes. Certaines d'entre-elles ont déjà été étudiées plus haut.

I-E-1 - BuildPath

La fonction **BuildPath** crée un chemin valide à partir d'un nom de dossier et d'un nom de fichier.

Syntaxe :

```
BuildPath(Path As String, Name As String) As String
```

Exemple :

```
Dim oFSO As Scripting.FileSystemObject
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
MsgBox oFSO.BuildPath("D:\Essai", "monFichier.txt")
```

Retourne : D:\Essai\monFichier.txt

I-E-2 - GetDriveName, GetFileName, GetBaseName, GetExtensionName

Ces fonctions extraient respectivement le disque, le nom complet, le nom et l'extension d'un fichier à partir du chemin passé en paramètre.

Exemple :

```
Dim oFSO As Scripting.FileSystemObject
Dim strChemin As String
strChemin = "D:\Essai\monfichier.txt"
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Debug.Print oFSO.GetDriveName(strChemin)
Debug.Print oFSO.GetBaseName(strChemin)
Debug.Print oFSO.GetFileName(strChemin)
Debug.Print oFSO.GetExtensionName(strChemin)
```

Retour :

```
D:
monfichier
monfichier.txt
txt
```

I-E-3 - GetParentFolder

La fonction **GetParentFolder** retourne le chemin d'accès du répertoire parent d'un fichier (ou d'un dossier).

Exemple :

```
Debug.Print oFSO.GetParentFolderName("D:\Essai\
```

Retour :

```
D:\Essai
```

I-E-4 - GetTempName

La fonction **GetTempName** retourne un nom de fichier temporaire depuis le système d'exploitation.

Exemple :

```
Dim oFSO As Scripting.FileSystemObject
Dim strChemin As String
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Debug.Print oFSO.GetTempName
```

Retour :

```
rad6AE46.tmp
```



Les fichiers temporaires sont utilisés pour stocker et traiter des données sur le disque dur de l'ordinateur. L'utilisation de cette fonction permet d'obtenir un nom de fichier libre.

I-E-5 - Tester l'existence d'un chemin

Bien que le **FileSystemObject** offre plusieurs méthodes pour tester l'existence d'un dossier, il est difficile en cas d'erreur de savoir quelle partie du chemin pose problème en cas d'erreur. Imaginons ce chemin :

D:\Essai\Essai2\Essai3

Essai2 n'existe pas. **FolderExists** renverra faux mais l'utilisateur ignorera si c'est à cause d'*Essai*, *Essai2*, ou *Essai3*.

Je vous propose la fonction ci-dessous plus complète permettant d'identifier le dossier à l'origine de l'erreur.

```
Function TestChemin(strChemin As String) As Boolean
On Error GoTo err
Dim oFSO As Scripting.FileSystemObject
Dim oFld As Scripting.Folder
Dim oDrv As Scripting.Drive
Dim strDossiers() As String
Dim i As Integer
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject

'Accède au disque
Set oDrv = oFSO.Drives(oFSO.GetDriveName(strChemin))
'Instancie le dossier racine
Set oFld = oDrv.RootFolder
'Découpe le chemin en dossier
strDossiers = Split(strChemin, "\")
'Tente d'accéder à chaque sous dossier
For i = 1 To UBound(strDossiers) - 1
Set oFld = oFld.SubFolders(strDossiers(i))
Next i
TestChemin = True
fin:

Exit Function

err:
Select Case err.Number
Case 5: MsgBox "Le disque n'existe pas"
Case 76: MsgBox "Impossible de trouver le dossier : " & strDossiers(i)
Case Else: MsgBox "Erreur inconnue"
```

```
End Select  
Resume fin  
End Function
```

II - Les fichiers textes

II-A - Introduction

Les fichiers textes sont constituées de lignes séparées par deux caractères : retour chariot et changement de lignes : **VbCrLf**

Deux techniques peuvent être utilisées pour lire et écrire dans de tels fichiers. La première est historique via la méthode **Open** de VB. La seconde fait appel à la librairie **Microsoft Scripting Runtime** et plus particulièrement ses objets **TextStream**.

II-B - Accès séquentiel

Ce type d'accès est historique dans le sens où pendant longtemps il était le seul disponible en Visual Basic. Le fichier est ouvert via la méthode **Open** et est référencé ensuite par un numéro de fichier.

Trois modes d'ouverture sont disponibles :

- **Input** : lecture seule
- **Output** : écriture (le fichier est vidé à chaque ouverture)
- **Append** : écriture en ajout

Exemple :

```
Open "D:\essai\monfichier.txt" For Input As 1
```

1 correspond ici au numéro de fichier. Un même numéro ne peut être réutilisé tant que le fichier qu'il référence n'est pas fermé. Pour remédier à ce problème et obtenir un numéro de fichier toujours valide, la fonction **Freefile** peut être utilisée.

```
Dim intFic As Integer  
intFic = FreeFile  
Open "D:\essai\monfichier.txt" For Input As intFic
```

Les numéros de fichier entre 1 et 255 donnent un accès exclusif au fichier. Il ne peut pas être ouvert par une autre application. En revanche, un numéro entre 256 et 511 donne un accès partagé.

FreeFile sans paramètre retourne un numéro inférieur à 255. Pour obtenir un numéro compris entre 256 et 511, il faut utiliser : FreeFile (1)

Quelque soit le mode d'ouverture, la fermeture se fait toujours via la méthode **Close**.

```
Close intFic
```

II-B-1 - Lecture

La lecture d'une ligne se fait via la méthode **Line Input #**. Le contenu est alors envoyé dans une variable de type **String**.

Exemple :

```
Dim intFic As Integer
Dim strLigne As String
intFic = FreeFile
Open "D:\essai\monfichier.txt" For Input As intFic
Line Input #intFic, strLigne
MsgBox strLigne
Close intFic
```

Cet exemple ne lit qu'une seule ligne : la première. Pour lire la totalité du fichier, il faut itérer l'opération jusqu'à la fin du fichier. La fonction **EOF (End Of File)** permet de savoir si la fin du fichier a été atteinte.

```
Dim intFic As Integer
Dim strLigne As String
intFic = FreeFile
Open "D:\essai\monfichier.txt" For Input As intFic
While Not EOF(intFic)
Line Input #intFic, strLigne
MsgBox strLigne
Wend
Close intFic
```

Comme vous pouvez le constater, il n'y a pas de notion de « **movenext** ». Le passage à la ligne suivante est géré automatiquement par **Line Input #**.

II-B-2 - Ecriture

L'écriture dans un fichier ouvert en mode **Output** ou **Append** est assurée par la fonction **Print #**.

Exemple :

```
Dim intFic As Integer

intFic = FreeFile
Open "D:\essai\monfichier.txt" For Output As intFic
Print #intFic, "Une ligne"
Close intFic
```

L'appel de **Print #** place automatiquement **VbCrLf** à la fin de la ligne à chaque appel.

Rappel : En mode **Output**, le contenu du fichier est écrasé. Si vous souhaitez ajouter votre texte en fin de fichier, utilisez le mode **Append**.

II-C - Accès direct

L'accès direct est très proche de l'accès séquentiel. La différence se situe au niveau de la structure du fichier. Dans le premier cas, nous avons un fichier où chaque ligne était une phrase. L'accès direct est réservé au fichier de données. Ce genre de fichier peut être obtenu lorsque vous exportez une table Access dans un fichier texte. Il s'agit en fait d'une suite d'enregistrements où chaque colonne possède une longueur fixe.

Par exemple :

Numéro du client sur 3 caractères, Nom sur 10 caractères, Prénom sur 10 autres caractères.

001NORD	PAUL	002MARIE	ANNE
---------	------	----------	------

Le fichier sera ouvert en mode **Random**. La fonction **Get** permet de lire un enregistrement et d'envoyer son contenu dans une variable. Pour que cela fonctionne correctement, il faut que la variable possède la même structure que l'enregistrement. Cette structure est obtenue à l'aide du bloc **Type**.

```
Type tClient
    Num As String * 3
    Nom As String * 10
    Prenom As String * 10
end type
```

La fonction **Get** permet de définir quel enregistrement lire. On peut par exemple lui indiquer de lire le deuxième client. Pour cela, il faut que VBA sache à quel octet commence l'enregistrement 2 dans le fichier. Le mot clé **Len** spécifié dans l'instruction **Open** permettra de définir la taille d'un enregistrement. Dès lors, en connaissant cette taille, **Get** sera capable de savoir où commencer sa lecture.

Exemple :

```
Dim intFic As Integer
Dim client As tClient
intFic = FreeFile
Open "D:\essai\monfichier.txt" For Random As intFic Len = Len(client)
Get intFic, 1, client
MsgBox client.Num
Get intFic, 2, client
MsgBox client.Num
Close intFic
```

Ici, chaque enregistrement possède la taille de la variable client (définie par la structure Type tclient).

Quant à l'écriture, elle se fait de la même manière avec l'instruction **Put** :

```
Dim intFic As Integer
Dim client As tClient
intFic = FreeFile
Open "D:\essai\monfichier.txt" For Random As intFic Len = Len(client)
'Fixe les informations du client
With client
    .Num = "003"
    .Nom = "JEAN"
    .Prenom = "PAUL"
End With
'Ecrit la donnée
Put intFic, 3, client
Close intFic
```

En spécifiant un numéro d'enregistrement déjà utilisé, vous écrasez la donnée précédente. Pour être certain d'ajouter un nouvel enregistrement, il faut que l'indice soit égal au nombre d'enregistrements déjà présents + 1. La fonction **LOF (Length Of File)** retourne la taille du fichier en octets. En divisant cette taille par la longueur de chaque enregistrement, on obtient aisément le nombre d'éléments.

L'ajout devient alors :

```
Dim intFic As Integer
Dim client As tClient
Dim intNum As Integer
intFic = FreeFile
Open "D:\essai\monfichier.txt" For Random As intFic Len = Len(client)
'Fixe les informations du client
With client
    .Num = "003"
    .Nom = "JEAN"
    .Prenom = "PAUL"
End With
```

```
'Compte le nombre d'enregistrements
intNum = LOF(intFic) / Len(client)
'Ajoute la donnée
Put intFic, intNum + 1, client
Close intFic
```

II-D - Les TextStream

Vous l'avez sans doute remarqué, la manipulation de fichier texte en VBA est très limitée. Pour combler ce manque, les développeurs ont souvent accès aux objets **TextStream** de la librairie **Microsoft Scripting Runtime**.

Même si l'accès est toujours séquentiel, la programmation est grandement simplifiée et paraît plus naturelle.

II-D-1 - Ouvrir un fichier texte

La méthode **OpenTextFile** de l'objet **FileSystemObject** permet d'ouvrir ou créer un fichier texte.

Syntaxe :

```
OpenTextFile(fileName As String, [IOMode As IOMode = ForReading], [Create As Boolean = Faux],
[Format As Tristate = TristateFalse]) As TextStream
```

- Le paramètre **FileName** correspond au chemin du fichier.
- **IOMode** spécifie le mode d'ouverture : **ForReading** (lecture), **ForWriting** (écriture), **ForAppending** (Ajout)
- Lorsque **Create** est à **True**, le fichier est créé s'il n'existe pas.
- Si **Format** est à **True**, le texte est en Unicode, sinon, il s'agit du codage ASCII.

Exemple :

```
Dim oFSO As Scripting.FileSystemObject
Dim oTxt As Scripting.TextStream
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Set oTxt = oFSO.OpenTextFile("D:\Essai\monfichier.txt", ForReading)
```



Une nouvelle fois, il faut choisir entre lecture et écriture. Impossible d'utiliser les deux modes simultanément.

La méthode **CreateTextFile** peut aussi être utilisée si le fichier n'existe pas encore.

Syntaxe :

```
CreateTextFile(fileName As String, [Overwrite As Boolean = Vrai], [Unicode As Boolean = Faux]) As
TextStream
```

Dans ce cas, le paramètre **OverWrite** écrase un fichier existant s'il est égal à **True**.

Une dernière technique peut être utilisée pour instancier un objet **TextStream** : utiliser la méthode **OpenAsTextStream** depuis un objet **File**.

Exemple :

```

Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Dim oTxt As Scripting.TextStream
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Set oFl = oFSO.GetFile("D:\Essai\monfichier.txt")
Set oTxt = oFl.OpenAsTextStream(ForReading)

```

II-D-2 - Lecture

La lecture d'un fichier texte se fait en positionnant le paramètre **IOMode** à **ForReading** lors de l'ouverture.

Plusieurs méthodes permettent de lire un ensemble de caractères :

- **Read(X)** : lit X caractères
- **ReadLine** : lit toute la ligne
- **ReadAll** : lit tout le fichier

La lecture se fait à partir de la position courante. Celle-ci est recalculée à chaque appel d'une de ces méthodes. Les propriétés **Line** et **Column** permettent de la connaître. Si à l'ouverture vous lisez 3 caractères, la position courante devient :

- line(ligne) = 1
- column(colonne) = 4

La propriété **AtEndOfLine** permet de savoir si vous êtes à la fin de la ligne. Quant à la propriété **AtEndOfStream**, elle détermine si vous êtes à la fin du fichier.

Exemple pour une lecture ligne à ligne :

```

Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Dim oTxt As Scripting.TextStream
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Set oFl = oFSO.GetFile("D:\Essai\monfichier.txt")
Set oTxt = oFl.OpenAsTextStream(ForReading)
With oTxt
    While Not .AtEndOfStream
        MsgBox .ReadLine
    Wend
End With

```

La méthode **Skip(X)** permet de déplacer la position courante de X caractères. La méthode **SkipLine** permet de sauter une ligne.


Exemple pour une lecture un caractère sur deux :

```

Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Dim oTxt As Scripting.TextStream
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Set oFl = oFSO.GetFile("D:\Essai\monfichier.txt")
Set oTxt = oFl.OpenAsTextStream(ForReading)
With oTxt
    While Not .AtEndOfStream
        MsgBox .Read(1)
        .Skip 1
    Wend
End With

```

```
End With
```

 *N'oubliez pas que les sauts de lignes sont aussi des caractères.*

II-D-3 - Ecriture

Que le fichier soit ouvert en mode **ForAppending** ou **ForWriting** n'influe pas sur les méthodes à utiliser :

- **Write(Text)** : écrit le texte à partir de la position courante
- **WriteLine(Text)** : écrit le texte dans une nouvelle ligne
- **WriteBlankLine(X)** : écrit X lignes vides.

Exemple :

```
Dim oFSO As Scripting.FileSystemObject
Dim oFl As Scripting.File
Dim oTxt As Scripting.TextStream
Dim i As Integer
'Instanciation du FSO
Set oFSO = New Scripting.FileSystemObject
Set oFl = oFSO.GetFile("D:\Essai\monfichier.txt")
Set oTxt = oFl.OpenAsTextStream(ForWriting)
With oTxt
    For i = 0 To 10
        .WriteLine i
    Next i
End With
```

II-D-4 - Fermeture

La fermeture d'un objet **TextStream** est simplement obtenue avec l'appel de sa méthode **Close**.

Exemple :

```
Txt.Close
```

III - La recherche de fichiers

La recherche de fichiers peut se faire de plusieurs manières. Sans appel à une référence externe, il faut utiliser la méthode **Dir** de VBA. Malheureusement, c'est très rudimentaire. Je vous propose deux types de recherche :

- la recherche récursive en utilisant le **FileSystemObject**
- la recherche optimisée en utilisant l'objet **FileSearch Office**

III-A - Recherche récursive en utilisant le FSO

L'idée : parcourir chaque dossier depuis la racine du disque. Vérifier si le fichier recherché est présent. Répéter l'opération pour chacun des sous-dossiers.

Un exemple possible :

```
Sub Explorer(p_strFichier As String, p_strCheminDepart As String, Optional p_oFld As Scripting.Folder)
On Error GoTo err
    Dim oFSO As Scripting.FileSystemObject
    Dim oFld As Scripting.Folder
    Dim oFl As File
    If p_oFld Is Nothing Then
        'Instanciation du FSO
        Set oFSO = New Scripting.FileSystemObject
        'Accède au répertoire du départ de recherche
        Set p_oFld = oFSO.GetFolder(p_strCheminDepart)
    End If
    Set oFl = p_oFld.Files(p_strFichier)
    MsgBox oFl.Path

SubDir:
'Explore les sous-dossiers
For Each oFld In p_oFld.SubFolders
    Explorer p_strFichier, p_strCheminDepart, oFld
DoEvents
Next oFld

fin:
Exit Sub
err:
Select Case err.Number
Case 53: Resume SubDir
Case Else:
    MsgBox "Erreur inconnue"
    Resume fin
End Select

End Sub
```

Qui sera lancé par :

```
Explorer "monfichier.txt", "d:\"
```

III-B - L'objet FileSearch

Fournit par l'objet **Access.Application**, l'objet **FileSearch** permet de rechercher des documents sur n'importe quel disque de l'ordinateur. Il utilise le mécanisme de recherche propre à Windows ce qui en fait un outil beaucoup plus puissant que la recherche récursive vu précédemment.

Un objet **FileSearch** est retourné par la propriété **Access.Application**. L'ajout de la référence **Microsoft Office X**

Object Library n'est pas indispensable. Toutefois, si vous ne la déclarez pas, vous ne pourrez pas utiliser les constantes prédéfinies ni bénéficier de la fonction IntelliSense (Aide à la saisie)

III-B-1 - Recherche basique

Prenons l'exemple ci-dessous :

```
Dim oFS As Office.FileSearch
Set oFS = Application.FileSearch
With oFS
    .NewSearch
    .FileType = msoFileTypeAllFiles
    .FileName = "monfichier.txt"
    .LookIn = "D:\Essai"
    .Execute
    MsgBox .FoundFiles.Count
End With
```

Comme indiqué plus haut, l'objet est accessible via la propriété du même nom dans **Application**. La méthode **NewSearch** déclare une nouvelle recherche et réinitialise les critères à leur valeur par défaut. La propriété **FileType** permet de paramétrer le type des fichiers recherchés. Il est ainsi possible de restreindre la recherche uniquement aux fichiers Word, aux fichiers Access, etc. Le nom du fichier recherché est à affecter à la propriété **FileName**. **LookIn** spécifie le chemin du dossier dans lequel la recherche sera effectuée. En définissant **SearchSubFolders** à **True**, elle sera étendue aux sous-répertoires.

Appelez la méthode **Execute** pour démarrer la recherche. Les fichiers trouvés sont retournés sous la forme d'une collection de noms de fichiers : **FoundFiles**. Comme pour toute collection, la propriété **Count** donne le nombre d'éléments.

Voici un exemple de parcours de la collection :

```
Dim oFS As Office.FileSearch
Dim i As Integer
Set oFS = Application.FileSearch

With oFS
    .NewSearch
    .FileType = msoFileTypeAllFiles
    .FileName = "monfichier.txt"
    .LookIn = "D:\Essai"

    .Execute
    For i = 1 To .FoundFiles.Count
        MsgBox .FoundFiles(i)
    Next i
End With
```

III-B-2 - Recherche complexe

L'objet **FileSearch** offre des possibilités étendues par la définition de filtre. Il est possible d'associer plusieurs filtres à une recherche.

Soit le besoin suivant :

« Je recherche tous les fichiers Word ou Excel dans le répertoire Essai modifié entre le 12/10/06 et le 15/10/06 »

Avec les propriétés vues plus haut, il est impossible de traduire le critère sur la date.

Pour cela il faut faire appel à la collection **PropertyTests** de l'objet **FileSearch**. Chaque objet de cette collection correspond à un prédicat composé de :

- Un nom de propriété (**Name**)
- Sa valeur (**Value**), une seconde valeur optionnelle (**SecondValue**)
- Un opérateur (**Condition**)
- Un opérateur logique pour définir comment sera évalué ce critère par rapport aux autres.

En fait, la collection **PropertyTests** correspond à cet écran :

Menu Fichier, Ouvrir, Recherche Avancée

Ici le prédicat pour la date sera :

Nom de la propriété : Modifié le

Opérateur : **msoConditionAnytimeBetween** (Date comprise entre #)

Valeur : 12/10/06

Seconde Valeur : 15/10/2006



Dans l'écran affiché plus haut, il était affiché date avant le # et date après le#. VBA offre l'opérateur Date entre #

La méthode **Add** permet d'ajouter un objet **PropertyTest** :

```
With oFS
    .NewSearch
    .FileType = msoFileTypeExcelWorkbooks + msoFileTypeWordDocuments
    .LookIn = "D:\Essai"
    With .PropertyTests
        .Add Name:="Modifié le", _
            Condition:=msoConditionAnytimeBetween, _
            Value:="10/10/2006", SecondValue:="15/10/2006", _
            Connector:=msoConnectorAnd
    End With
    .Execute
    For i = 1 To .FoundFiles.Count
        MsgBox .FoundFiles(i)
    Next i
End With
```

La liste des propriétés et des opérateurs est très longue. Il est difficile de toutes les exposer ici. Je vous propose plutôt d'étudier les différentes valeurs des énumérations via l'**explorateur d'objets**. Leur traduction est très simple.

IV - Conclusion

Vous voilà maintenant prêt à faire face à la plupart des problèmes concernant les fichiers en VBA.

N'hésitez pas à consulter les pages sources [Access](#) et [Visual Basic](#), elles regroupent plusieurs exemples d'import de données via des fichiers textes.

Si vous ne parvenez pas à utiliser les éléments donnés dans ce document et avant de poser toute question sur le [forum Access](#), vérifiez que vous avez bien ajouté les références nécessaires. Il s'agit de :

- **Microsoft Scripting Runtime** pour le **FileSystemObject**
- **Microsoft Office X.0 Object Library** pour l'objet **FileSearch** (Recherche de fichiers)

Je tiens à remercier tous ceux qui ont participé à ce document de par leur relecture et leurs précieux conseils.