Un complément FileSearch pour Excel 2007

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 24/06/2007

Dernière mise à jour :

Ce tutoriel propose un complément Excel 2007 pour remplacer et personnaliser l'objet FileSearch.

I - Introduction

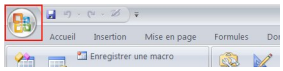
L'objet **FileSearch** n'étant plus supporté dans Office2007, cet article propose une solution de substitution pour Excel.
Le classeur xla, téléchargeable en bas de cette page, contient un module de classe ClasseFileSearch pour gérer la recherche de fichiers sur votre PC.
Lorsque le complément sera installé, vous pourrez rédiger facilement des macros qui rechercheront des fichiers à l'intérieur de répertoires déterminés.
Nota.
Même si l'utilisation n'a pas grand intérêt dans cette configuration, sachez que la procédure fonctionne aussi dans Excel2002.

II - Description

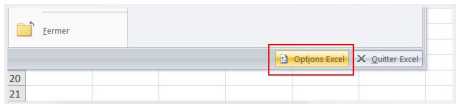
La procédure recherche des fichiers en fonction des critères spécifiés:
* Le répertoire contenant les fichiers à rechercher
* Option pour rechercher aussi dans les sous dossiers
* Option pour rechercher un type de fichier spécifique
* Option de tri

et renvoie le résultat dans un tableau contenant:
* Le nom des fichiers
* Le chemin
* La taille des fichiers (en octets)
* La date de création
* La date de dernière modification
* Le type de fichier

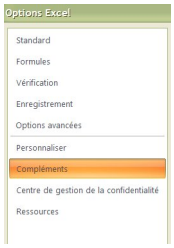
Installez la macro complémentaire dans le dossier qui leur ai réservé.
Les fichiers .xla Excel2007 sont généralement stockés dans le répertoire:
C:\Documents and Settings\NomUtilisateur\Application Data\Microsoft\AddIns
Ensuite, pour que le complément soit opérationnel à chaque ouverture de l'application:
Cliquez le bouton "**Office**".



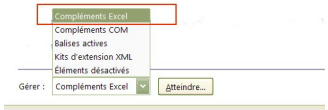
Cliquez sur le bouton "**Options Excel**".



Sélectionnez le menu "**Compléments**".



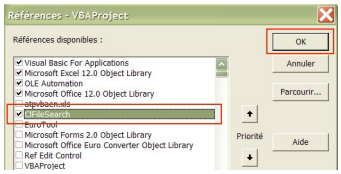
Choisissez "**Compléments Excel**" dans le menu déroulant "**Gérer**" (en bas de la fenêtre).



Cliquez sur le bouton "**Atteindre:**".
La fenêtre qui s'affiche est identique aux versions antérieures d'Excel.
Cochez le complément "**Classefilesearch**".
Nota:
Cliquez sur le bouton **Parcourir** si le complément n'apparaît pas dans la liste ou s'il est stocké dans un autre emplacement que celui prévu par l'application.

Cliquez sur le bouton "**OK**" pour valider.
Désormais, lorsque vous souhaitez utiliser cette fonction de recherche, il suffit d'activer la référence **ClFileSearch**:
Dans l'éditeur de macro,

Menu Outils
Références
Cochez la ligne "**ClFileSearch**".



Cliquez sur le bouton **OK** pour valider.
Remarque:
Si la référence **ClFileSearch** n'apparait pas dans la liste, fermez totalement l'application Excel puis ré-ouvrez la.
Et pour terminer, vous pouvez utiliser la syntaxe suivante dans vos projets:
Collez l'exemple ci-dessous dans un module standard et adaptez simplement le nom du répertoire contenant les fichiers à rechercher.
Le résultat est écrit dans la fenêtre d'exécution de l'éditeur VBA (**Ctrl+G**).

Vba

```
Sub Test()
'Nécessite d'activer la référence ClFileSearch
'(Dans l'éditeur de macros: Menu Outils/Références)
Dim i As Long
Dim Recherche As ClFileSearch.ClasseFileSearch


Set Recherche = ClFileSearch.Nouvelle_Recherche

With Recherche
    'Définit le répertoire de recherche
    .FolderPath = "C:\Documents and Settings\mimi\NomDossier"

    'Définit la recherche dans les sous dossiers (True / False)
    .SubFolders = False

    'Option de tri:
    '(Sort_None, sort_Name, sort_Path, sort_Size, sort_DateCreated, sort_LastModified, sort_Type)
    'Pas de tri si le paramètre n'est pas spécifié.
    .SortBy = sort_Name

    'Option pour rechercher un type de fichier
    '(Renvoie tous les fichiers si non spécifié)
    '.Extension = "*.doc"

    'Execute la recherche
    .Execute

    'Boucle sur le tableau pour afficher le résultat de la recherche
    '(.FoundFilesCount renvoie le nombre de fichiers trouvés)
    For i = 1 To .FoundFilesCount
        Debug.Print .Files(i).strFileName 'nom du fichier
        Debug.Print .Files(i).strPathName 'chemin
        Debug.Print .Files(i).lngSize & "  octets" 'taille
        Debug.Print .Files(i).DateCreated 'date création fichier
        Debug.Print .Files(i).DateLastModified 'date dernière modification
        Debug.Print .Files(i).strFileType 'type de fichier

        Debug.Print "---"
    Next
End With

Set Recherche = Nothing

End Sub
```

Les options de tri et de recherche par extension sont facultatives.
Si vous ne spécifiez pas l'option de tri, la recherche sera un peu plus rapide.

Utilisez un des paramètres suivant pour définir sur quelle colonne du tableau de résultat (**TabFiles**) doit être appliqué le tri:

- sort_Name: Tri sur le nom des fichiers
- sort_Path: Tri sur le nom des répertoires
- sort_Size: Tri sur la taille des fichiers
- sort_DateCreated: Tri sur la date de création des fichiers
- sort_LastModified: Tri sur la date de dernière modification des fichiers
- sort_Type: Tri sur les types de fichiers

Lorsque vous appliquez un tri, For i = 1 To .FoundFilesCount affiche le résultat par ordre croissant.
Utilisez For i = .FoundFilesCount To 1 Step -1 pour afficher le résultat par ordre décroissant.

III - Les procédures du complément

Dans un module Standard:

Vba

```
Option Explicit

Public Type InfosResultFichiers
    strFileName As String
    strPathName As String
    lngSize As Long
    DateCreated As Date
    DateLastModified As Date
    strFileType As String
End Type
'-----------------------------------------
```

Vba

```
Public Function Nouvelle_Recherche() As ClasseFileSearch
    Set Nouvelle_Recherche = New ClasseFileSearch
End Function
```

Dans un module de classe nommé **ClasseFileSearch**:

Vba

```
Option Explicit
Option Compare Text
Option Base 1


'-------------------------------------------------

'Module de classe ClasseFileSearch pour Excel 2007
'SilkyRoad
'http://silkyroad.developpez.com/
'
'
'Mise à jour le 01.07.2007

'-------------------------------------------------


'La procédure recherche des fichiers en fonction des critères
'spécifiés et renvoie dans un tableau :

    'Le nom des fichiers
    'Le chemin
    'La taille des fichers (en octets)
    'La date de création
    'La date de dernière modification
    'Le type de fichier)


'-------------------------------------------------


'Enumération pour les options de tri
Public Enum Sort_By
    Sort_None
    sort_Name
    sort_Path
    sort_Size
    sort_DateCreated
    sort_LastModified
```

Vba

```
    sort_Type
End Enum


Dim TabFiles() As InfosResultFichiers
Dim DirectoryPath As String
Dim lngFoundFilesCount As Long
Dim boolSousRep As Boolean
Dim strExtens As String
Dim optionSortBy As Long



'Propriété pour le répertoire de recherche
Public Property Let FolderPath(strFolderPath As String)
    DirectoryPath = strFolderPath
End Property


'Propriété pour rechercher dans les sous dossiers
Public Property Let SubFolders(boolSubFolders As Boolean)
    boolSousRep = boolSubFolders
End Property


'Propriété pour lister les fichiers correspondants à la requête
Public Property Get Files(Idx As Long) As InfosResultFichiers
    Files = TabFiles(Idx)
End Property


'Propriété pour l'extension des fichiers à rechercher
Public Property Let Extension(strExtension As String)
    strExtens = strExtension
End Property


'Propriété pour compte le nombre de fichiers
Public Property Get FoundFilesCount() As Long
    FoundFilesCount = lngFoundFilesCount
End Property


'Propriété pour l'option de tri
Public Property Let SortBy(lngSortBy As Sort_By)
    optionSortBy = lngSortBy
End Property


'Fonction d'exécution
Public Function Execute() As Long
    'Lance la recherche
    ListeFichiers DirectoryPath

    'Vérifie que des fichiers ont été trouvés et qu'une option de tri a
    'été spécifié avant de lancer la procédure de tri.
    If lngFoundFilesCount > 1 And optionSortBy <> Sort_By.Sort_None Then _
        FonctionTri optionSortBy

    Execute = lngFoundFilesCount
End Function



'Procédure pour lister les fichiers
Private Sub ListeFichiers(strFolderName As String)
    Dim Fso As Object
    Dim NomDossier As Object, SousDossier As Object
    Dim objFichier As Object

    On Error GoTo Fin
```

Vba

```
    'Vérifie si le dossier spécifié existe
    If Dir(strFolderName, vbDirectory Or vbHidden Or vbSystem) = "" Then Exit Sub

    Set Fso = CreateObject("Scripting.FileSystemObject")
    Set NomDossier = Fso.GetFolder(strFolderName)


    'Boucle sur les fichiers du répertoire
    For Each objFichier In NomDossier.Files

        'Vérifie l'extension du fichier
        If objFichier.Name Like strExtens Or strExtens = "" Then

            'Redimensionne le tableau pour ajouter un nouvel élément
            lngFoundFilesCount = lngFoundFilesCount + 1
            ReDim Preserve TabFiles(lngFoundFilesCount)

            'Nom fichier
            TabFiles(lngFoundFilesCount).strFileName = objFichier.Name
            'Répertoire
            TabFiles(lngFoundFilesCount).strPathName = objFichier.ParentFolder
            'Taille du fichier (en octets)
            TabFiles(lngFoundFilesCount).lngSize = objFichier.Size
            'Date de création
            TabFiles(lngFoundFilesCount).DateCreated = objFichier.DateCreated
            'Date de création ou dernière modification
            TabFiles(lngFoundFilesCount).DateLastModified = objFichier.DateLastModified
            'Type de fichier
            TabFiles(lngFoundFilesCount).strFileType = objFichier.Type
        End If
    Next objFichier


    'Boucle récursive:
    '(Si l'option de recherche dans les sous répertoires a été spécifiée)
    If boolSousRep Then
        For Each SousDossier In NomDossier.SubFolders
            ListeFichiers SousDossier.Path
        Next SousDossier
    End If


Exit Sub:

Fin:
MsgBox "Erreur '" & Err.Number & "'" & vbCrLf & vbCrLf & _
    Err.Description, vbInformation
End Sub



'Procédure de tri (reste à améliorer).
Private Sub FonctionTri(optionSortBy As Sort_By)
    Dim i As Long, j As Long, k As Long
    Dim ValTemp As Variant

    'Vérifie quel champ du tableau doit être trié
    Select Case optionSortBy

        Case Sort_By.sort_Name
            For i = LBound(TabFiles) To UBound(TabFiles)
                j = i
                For k = j + 1 To UBound(TabFiles)
                    If TabFiles(k).strFileName <= TabFiles(j).strFileName Then j = k
                    If TabFiles(k).strFileName <= TabFiles(j).strFileName Then j = k
                Next k

                If i <> j Then
                    ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
```

Vba

```
                    TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                    TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                    TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                    TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp

                ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                    TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                    TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
            End If
        Next i


    Case Sort_By.sort_Path
        For i = LBound(TabFiles) To UBound(TabFiles)
            j = i
            For k = j + 1 To UBound(TabFiles)
                If TabFiles(k).strPathName <= TabFiles(j).strPathName Then j = k
                If TabFiles(k).strPathName <= TabFiles(j).strPathName Then j = k
            Next k

            If i <> j Then
                ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
                    TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                    TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                    TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                    TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp

                ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                    TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                    TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
            End If
        Next i


    Case Sort_By.sort_Size
        For i = LBound(TabFiles) To UBound(TabFiles)
            j = i
            For k = j + 1 To UBound(TabFiles)
                If TabFiles(k).lngSize <= TabFiles(j).lngSize Then j = k
                If TabFiles(k).lngSize <= TabFiles(j).lngSize Then j = k
            Next k

            If i <> j Then
                ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
                    TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                    TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                    TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                    TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp
```

Vba

```
                    ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                        TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                    ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                        TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
                End If
            Next i


        Case Sort_By.sort_DateCreated
            For i = LBound(TabFiles) To UBound(TabFiles)
                j = i
                For k = j + 1 To UBound(TabFiles)
                    If TabFiles(k).DateCreated <= TabFiles(j).DateCreated Then j = k
                    If TabFiles(k).DateCreated <= TabFiles(j).DateCreated Then j = k
                Next k

                If i <> j Then
                    ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
                        TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                     ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                        TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                    ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                        TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                     ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                        TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp

                    ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                        TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                    ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                        TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
                End If
            Next i


        Case Sort_By.sort_LastModified
            For i = LBound(TabFiles) To UBound(TabFiles)
                j = i
                For k = j + 1 To UBound(TabFiles)
                    If TabFiles(k).DateLastModified <= TabFiles(j).DateLastModified Then j = k
                    If TabFiles(k).DateLastModified <= TabFiles(j).DateLastModified Then j = k
                Next k

                If i <> j Then
                    ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
                        TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                     ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                        TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                    ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                        TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                     ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                        TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp

                    ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                        TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                    ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                        TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
                End If
            Next i

         Case Sort_By.sort_Type
            For i = LBound(TabFiles) To UBound(TabFiles)
                j = i
```

Vba
```
                For k = j + 1 To UBound(TabFiles)
                    If TabFiles(k).strFileType <= TabFiles(j).strFileType Then j = k
                    If TabFiles(k).strFileType <= TabFiles(j).strFileType Then j = k
                Next k

                If i <> j Then
                    ValTemp = TabFiles(j).strFileName: TabFiles(j).strFileName = _
                        TabFiles(i).strFileName: TabFiles(i).strFileName = ValTemp

                     ValTemp = TabFiles(j).strPathName: TabFiles(j).strPathName = _
                        TabFiles(i).strPathName: TabFiles(i).strPathName = ValTemp

                    ValTemp = TabFiles(j).lngSize: TabFiles(j).lngSize = _
                        TabFiles(i).lngSize: TabFiles(i).lngSize = ValTemp

                      ValTemp = TabFiles(j).DateCreated: TabFiles(j).DateCreated = _
                        TabFiles(i).DateCreated: TabFiles(i).DateCreated = ValTemp

                    ValTemp = TabFiles(j).DateLastModified: TabFiles(j).DateLastModified = _
                        TabFiles(i).DateLastModified: TabFiles(i).DateLastModified = ValTemp

                    ValTemp = TabFiles(j).strFileType: TabFiles(j).strFileType = _
                        TabFiles(i).strFileType: TabFiles(i).strFileType = ValTemp
                End If
            Next i

    End Select
End Sub
```

Pour que le module de classe puisse être utilisé dans un classeur autre que celui dans lequel il est déclaré, j'ai utilisé le mode opératoire préconisé par Tushar Mehta:

Consultez l'article sur le site Microsoft.

La fonction de recherche des fichiers nécessite la présence sur votre poste de la DLL scrrun.dll (Microsoft Scripting Runtime).

IV - Conclusion

La source présentée sur cette page est un simple exemple, facilement adaptable aux spécificités de vos projets personnels.
Vous pourrez par exemple:
Personnaliser les données à insérer dans le tableau de résultat.
Améliorer la fonction de tri.
Utiliser d'autres méthodes pour lister les fichiers et récupérer des informations sur les fichiers.
Vos remarques, critiques et propositions d'amélioration sont les bienvenues.

V - Liens

Alternative zum Filesearch
How to use a class (object) from outside of the VBA project in which it is declared.
La manipulation des fichiers en VBA, par Christophe WARIN.

VI - Téléchargement

Téléchargez le complément.