

Lire et modifier les propriétés des classeurs et autres fichiers

par [SilkyRoad](#)

Date de publication : 17/08/2006

Dernière mise à jour : 03/03/2007

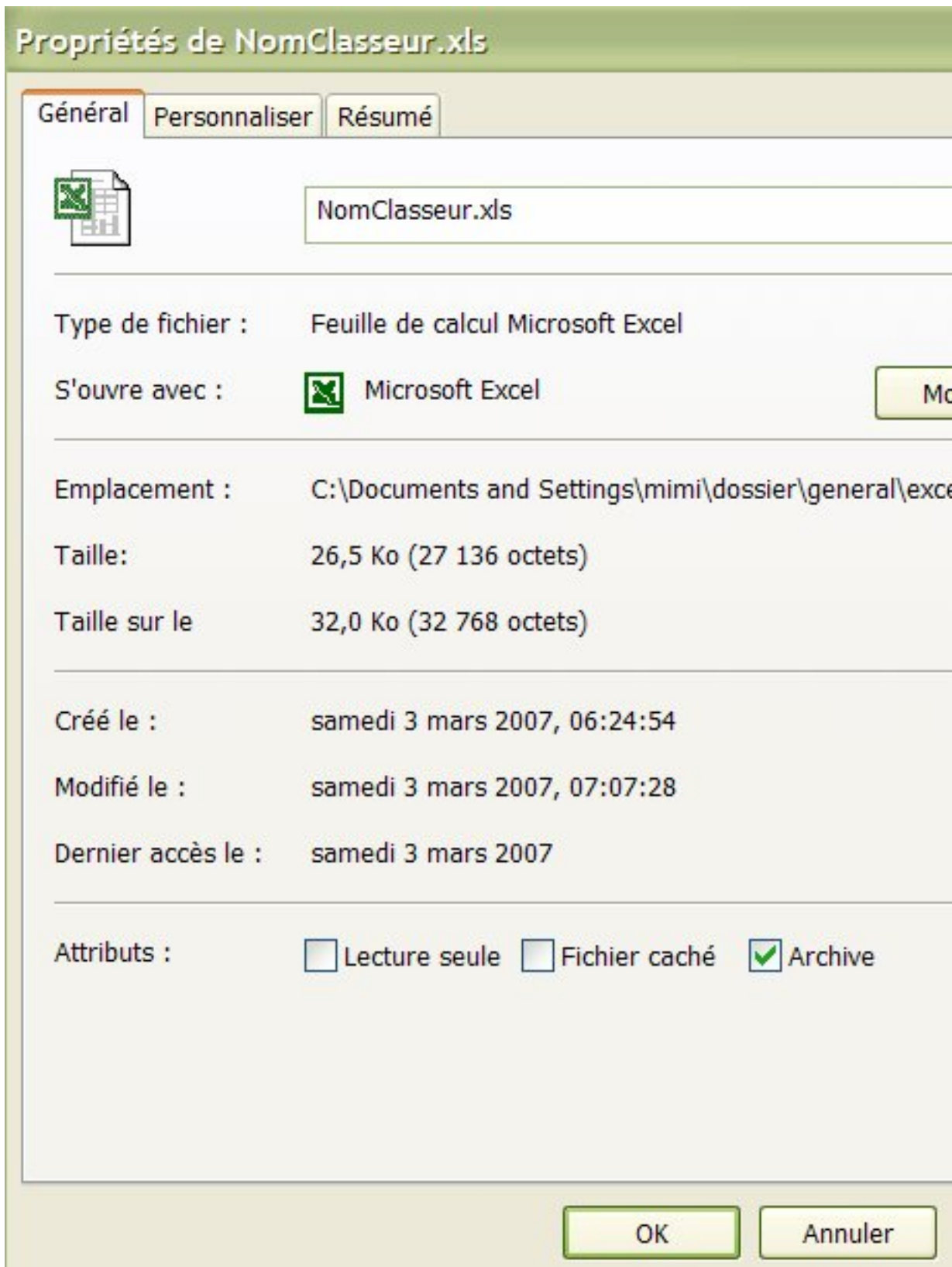
Ce document présente quelques méthodes pour lire et modifier les propriétés des Classeurs Excel. Certains des exemples proposés sont utilisables pour d'autres types de fichiers. Toutes les procédures ont été testées en utilisant Excel2002.

- I - Introduction
- II - Les propriétés d'un classeur ouvert
 - II-A - Lire les propriétés prédéfinies
 - II-B - Modifier les propriétés prédéfinies
 - II-C - Ajouter une propriété personnalisée
 - II-D - Lire Les propriétés personnalisées
 - II-E - Modifier une propriété personnalisée
 - II-F - Supprimer une propriété personnalisée
- III - Les propriétés d'un classeur fermé (Utilisation de la bibliothèque DSO)
 - III-A - Afficher les propriétés prédéfinies d'un classeur
 - III-B - Modifier les propriétés d'un classeur
 - III-C - Ajouter une propriété personnalisée dans un classeur
 - III-D - Lire une propriété personnalisée
- IV - Lire les propriétés de tous types de fichiers
 - IV-A - Utiliser la bibliothèque Microsoft Scripting Runtime (FileSystemObject)
 - IV-B - La fonction FileDateTime
 - IV-C - Lister les propriétés avancées
 - IV-D - Utiliser La classe CIM_DataFile
- V - Téléchargement

I - Introduction

Tous les types de fichiers possèdent des propriétés. Vous pouvez les visualiser depuis [l'explorateur Windows](#) en faisant un clic droit sur le fichier puis en choisissant l'option "**Propriétés**". En fonction du type de fichier sélectionné, un ou plusieurs onglets sont visibles dans la boîte de dialogue (Général, Personnaliser, Résumé, Version, Compatibilité, Signatures numériques, Récapitulatif, Statistiques...). Chaque onglet renvoie des informations spécifiques. L'onglet "Général" est commun pour tous les types de fichiers.

Les fichiers Office disposent des onglets Général, Personnaliser et Résumé.

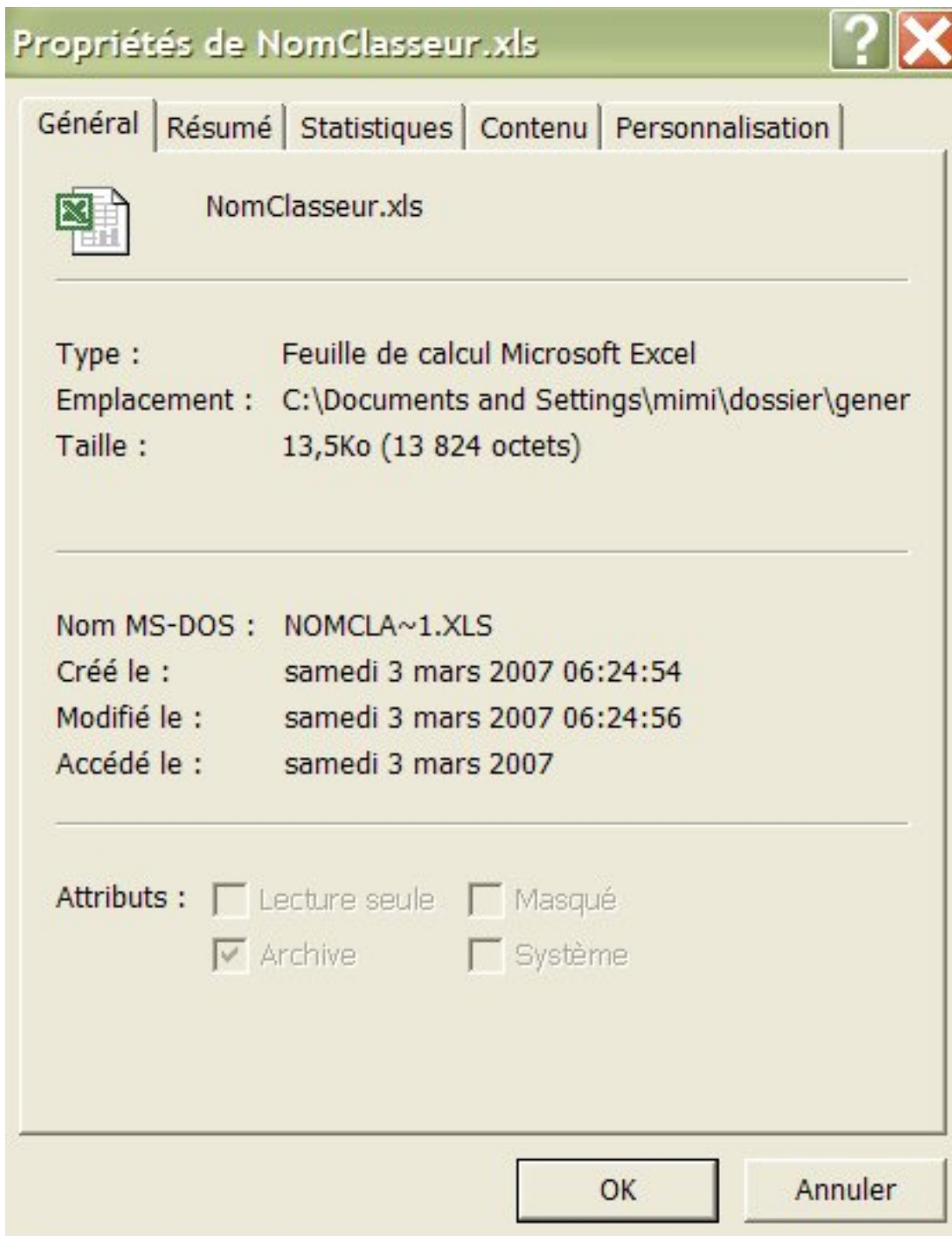


Ce document présente quelques méthodes pour lire et modifier les informations contenues dans ces onglets.

II - Les propriétés d'un classeur ouvert

II-A - Lire les propriétés prédéfinies

Les propriétés prédéfinies sont accessibles en utilisant le Menu principal d'Excel: Fichier/Propriétés.



Vous pouvez consulter (ou modifier pour certains d'entre eux) les champs contenus dans les différents onglets.

- * Général
- * Résumé
- * Statistiques
- * Contenu
- * Personnalisation

Les propriétés contiennent des informations permettant d'identifier un fichier (Titre descriptif, nom de l'auteur, le sujet et les mots clés qui identifient des rubriques ou d'autres informations importantes dans le fichier). Utilisez les propriétés d'un document pour afficher les informations relatives à un fichier ou organiser vos fichiers de manière à les retrouver facilement par la suite. Vous pouvez également rechercher des documents en fonction de leurs propriétés.

* Les propriétés automatiquement mises à jour comprennent les statistiques qui sont conservées par vos applications Microsoft Office, telles que la taille du fichier et les dates de création et de dernière modification des fichiers.

* Les champs des propriétés prédéfinies existent déjà (Auteur, Titre, Sujet, Commentaire...), mais vous devez entrer un texte libre pour compléter les champs.

* Les propriétés personnalisées sont des propriétés que vous définissez. Vous pouvez affecter un texte, une heure ou une valeur numérique aux propriétés personnalisées, ainsi que les valeurs booléennes. Vous pouvez faire un choix dans une liste de noms suggérés ou définir vos propres noms.

Si vous souhaitez afficher automatiquement la boîte de dialogue **Propriétés** lors de l'enregistrement des classeurs, afin de compléter les champs d'information:

Menu Outils/Options/Onglet "Général"/Cochez l'option "Afficher la fenêtre des propriétés."

Il est possible de lister par macro les propriétés prédéfinies d'un classeur, en utilisant la propriété **BuiltinDocumentProperties**.

Cet exemple permet de lister les propriétés du classeur actif dans une feuille de calcul.

Vba

```
Sub Test()  
  infosClasseurBuiltinDocumentProperties ActiveWorkbook  
End Sub  
  
Sub infosClasseurBuiltinDocumentProperties(Wb As Workbook)  
  Dim Valeur As DocumentProperty  
  Dim i As Byte  
  
  On Error Resume Next  
  
  'Boucle sur la collection de propriétés prédéfinies  
  For Each Valeur In Wb.BuiltinDocumentProperties  
    i = i + 1  
    'Insère le nom des propriétés dans la colonne A  
    ThisWorkbook.Worksheets(1).Cells(i, 1) = Valeur.Name  
    'Insère le contenu de la propriété dans la colonne B  
    ThisWorkbook.Worksheets(1).Cells(i, 2) = Valeur.Value  
  Next  
  
  ThisWorkbook.Worksheets(1).Columns("A:B").AutoFit  
End Sub
```

Un autre exemple pour lire une propriété particulière:

Vba

```
'Récupère le contenu de la propriété "Auteur"  
MsgBox ThisWorkbook.BuiltinDocumentProperties("Author").Value  
  
'Il est aussi possible d'appeler la propriété en utilisant l'index de la collection.  
'MsgBox ThisWorkbook.BuiltinDocumentProperties(3).Value
```

II-B - Modifier les propriétés prédéfinies

Si vous souhaitez modifier les propriétés par macro, utilisez:

Vba

```
'Modifie les commentaires  
ThisWorkbook.BuiltinDocumentProperties("Comments").Value = "Ajout commentaire"
```

II-C - Ajouter une propriété personnalisée

Il est possible d'ajouter une propriété personnalisée par macro.

Les propriétés personnalisées permettent de stocker les informations complémentaires de votre choix. Ces informations sont ensuite accessibles dans l'onglet "**Personnalisation**".

L'argument Type spécifie le type de données pour la nouvelle propriété. Attention: Vous obtiendrez un message d'erreur (Incompatibilité de type) si vous essayez d'insérer des données textes dans une propriété définie pour des valeurs numériques.

Les types de données possibles:

msoPropertyTypeNumber: Valeurs entières (Si vous insérez 196.4, c'est 196 qui sera enregistré)

msoPropertyTypeFloat: Valeurs numériques

msoPropertyTypeBoolean: Vrai ou Faux

msoPropertyTypeDate: Dates et heures

msoPropertyTypeString: Texte

Vba

```
Sub ajouterProprietePersonnalisee()  
    ThisWorkbook.CustomDocumentProperties.Add Name:="infoX", _  
        Type:=msoPropertyTypeNumber, LinkToContent:=False, Value:=1965  
End Sub
```

II-D - Lire Les propriétés personnalisées

Cet exemple permet de lire la propriété personnalisée qui a été créée par la macro précédente.

Vba

```
MsgBox ThisWorkbook.CustomDocumentProperties("infoX").Value
```

Il est aussi possible de boucler sur la collection de propriétés personnalisées:

Vba

```
Sub bouclerSurToutesLesProprietesPersonnalisees()  
    Dim Cp As DocumentProperty  
  
    'Vérifie qu'il y a des propriétés personnalisée  
    If ThisWorkbook.CustomDocumentProperties.Count = 0 Then Exit Sub  
  
    For Each Cp In ThisWorkbook.CustomDocumentProperties  
        MsgBox Cp.Name & vbLf & Cp.Value  
    Next Cp  
End Sub
```

II-E - Modifier une propriété personnalisée

Vba

```
ThisWorkbook.CustomDocumentProperties("infoX").Value = 1997
```

II-F - Supprimer une propriété personnalisée

Vba

```
ThisWorkbook.CustomDocumentProperties("infoX").Delete
```

Pour supprimer la collection de propriétés personnalisées, utilisez.

Vba

```
Sub SupprimeCollection_ProprietesPersonnalisees()  
    Dim Cst As DocumentProperty  
  
    For Each Cst In ThisWorkbook.CustomDocumentProperties  
        Cst.Delete  
    Next Cst  
End Sub
```

III - Les propriétés d'un classeur fermé (Utilisation de la bibliothèque DSO)

Les procédures présentées dans ce chapitre utilisent la bibliothèque **DSO oleDocument Properties Reader 2.0**

Si elle n'est pas installée sur ton poste , vous pouvez la télécharger sur le site de [Microsoft](#).

La librairie DSO oleDocument Properties Reader 2.0

Cette bibliothèque permet de lire et modifier les propriétés des documents Office (Excel, Word, Powerpoint...) sans les ouvrir.

III-A - Afficher les propriétés prédéfinies d'un classeur

```
Vb
Sub LireProprietesClasseur_DSO()
    'Nécessite d'activer la référence DSO OleDocument Properties Reader 2.0
    'http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q224351
    Dim DSO As DSOFfile.OleDocumentProperties

    Set DSO = New DSOFfile.OleDocumentProperties

    'Attention: Le fichier doit être préalablement fermé!
    DSO.Open sfilename:="C:\Documents and Settings\michel\leClasseur.xls"

    '
    MsgBox DSO.SummaryProperties.Author & vbLf & DSO.SummaryProperties.Comments
    '
    'Les autres propriétés:
    ' ApplicationName ' Author ' ByteCount ' Category ' CharacterCount
    ' CharacterCountWithSpaces ' Comments ' Company ' DateCreated
    ' DateLastPrinted ' DateLastSaved ' HiddenSlideCount
    ' Keywords ' LastSavedBy ' LineCount ' Manager ' MultimediaClipCount
    ' NoteCount ' PageCount ' ParagraphCount ' PresentationFormat
    ' RevisionNumber ' SharedDocument ' SlideCount
    ' Subject ' Template ' Title ' TotalEditTime ' Version ' WordCount
    '
    DSO.Close
End Sub
```

III-B - Modifier les propriétés d'un classeur

Voici un exemple pour modifier le champ "Commentaire".

```
Vb
Sub modifierProprietesClasseur_DSO()
    'Nécessite d'activer la référence DSO OleDocument Properties Reader 2.0
    'http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q224351
    Dim DSO As DSOFfile.OleDocumentProperties

    Set DSO = New DSOFfile.OleDocumentProperties

    'Attention: Le fichier doit être préalablement fermé!
    DSO.Open sfilename:="C:\Documents and Settings\michel\leClasseur.xls"
    DSO.SummaryProperties.Comments = "mon nouveau commentaire"
End Sub
```

Vb

```
DSO.Save
DSO.Close
End Sub
```

III-C - Ajouter une propriété personnalisée dans un classeur

Vb

```
Sub AjouterProprietePersonnalisee_DSO()
    Dim DSO As DSOFile.OleDocumentProperties

    Set DSO = New DSOFile.OleDocumentProperties

    'Attention: Le fichier doit être préalablement fermé!
    DSO.Open sfilename:="C:\Documents and Settings\michel\leClasseur.xls"
    DSO.CustomProperties.Add "maProprietePerso", "Bonjour"
    DSO.Save
    DSO.Close
End Sub
```

III-D - Lire une propriété personnalisée

Cet exemple permet de lire la propriété personnalisée qui a été créée par la macro précédente.

Vb

```
Sub LireProprietesPersonnalisees_DSO()
    Dim DSO As DSOFile.OleDocumentProperties

    Set DSO = New DSOFile.OleDocumentProperties

    'Attention: Le fichier doit être préalablement fermé!
    DSO.Open sfilename:="C:\Documents and Settings\michel\leClasseur.xls"
    MsgBox DSO.CustomProperties.Item("maProprietePerso").Value
    'Il est aussi possible de lire une propriété en utilisant
    'les index de la collection:
    'MsgBox DSO.CustomProperties.Item(0).Value
    DSO.Close
End Sub
```

IV - Lire les propriétés de tous types de fichiers

IV-A - Utiliser la bibliothèque Microsoft Scripting Runtime (FileSystemObject)

La bibliothèque Microsoft Scripting Runtime permet de récupérer les propriétés d'un fichier, principalement les informations qui apparaissent dans l'onglet "**Général**".

```
Vb
Sub Test()
    proprietesFichier_getFile "C:\Documents and Settings\michel\leClasseur.xls"
End Sub

Sub proprietesFichier_getFile(Fichier As String)
    'Nécessite d'activer la référence Microsoft Scripting Runtime
    Dim Cible As Scripting.FileSystemObject
    Dim Valeur As Scripting.File
    Dim Resultat As String

    Set Cible = CreateObject("Scripting.FileSystemObject")
    Set Valeur = Cible.GetFile(Fichier)

    Resultat = "Chemin et nom complet : " & Cible.GetAbsolutePathName(Valeur) & Chr(10) & Chr(10) & _
    "Chemin : " & Cible.GetParentFolderName(Valeur) & Chr(10) & Chr(10) & _
    "Nom fichier : " & Cible.GetFileName(Valeur) & Chr(10) & Chr(10) & _
    "Nom fichier sans extension : " & Cible.GetBaseName(Valeur) & Chr(10) & Chr(10) & _
    "Extension fichier : " & Cible.GetExtensionName(Valeur) & Chr(10) & Chr(10) & _
    "Chemin : " & Valeur.ParentFolder & Chr(10) & Chr(10) & _
    "Chemin court : " & Valeur.ShortPath & Chr(10) & Chr(10) & _
    "Nom court : " & Valeur.ShortName & Chr(10) & Chr(10) & _
    "Date creation : " & Valeur.dateCreated & Chr(10) & Chr(10) & _
    "Derniere modification : " & Valeur.dateLastModified & Chr(10) & Chr(10) & _
    "Taille fichier : " & Valeur.Size & " octets" & Chr(10) & Chr(10) & _
    "Type fichier : " & Valeur.Type

    MsgBox Resultat
End Sub
```

Cette bibliothèque permet aussi de passer un classeur en lecture seule, sans avoir besoin de l'ouvrir.

```
Vb
Sub passerClasseur_lectureSeule()
    'Nécessite d'activer la référence Microsoft Scripting Runtime
    Dim Fs As FileSystemObject
    Dim F As File

    Set Fs = CreateObject("Scripting.FileSystemObject")
    Set F = Fs.GetFile("C:\Documents and Settings\michel\leClasseur.xls")
    F.Attributes = F.Attributes + ReadOnly
End Sub
```

Et pour enlever la lecture seule, utilisez:

```
Vb
F.Attributes = F.Attributes + ReadOnly = False
```

IV-B - La fonction FileDateTime

La fonction **FileDateTime** renvoie la date et l'heure de création ou de dernière modification d'un fichier.

Vb

```
MsgBox FileDateTime("C:\Documents and Settings\michel\leClasseur.xls")
```

IV-C - Lister les propriétés avancées

Il est possible d'accéder à l'ensemble des propriétés avancées (onglet Résumé) en utilisant la bibliothèque "Microsoft Shell Controls and Automation".

[Le lien vers le site Microsoft](#)

Cet exemple boucle sur les fichiers d'un répertoire et affiche les propriétés.

Vb

```
Sub ListeProprietesFichiers_getDetailsOf()  
'source:  
'http://www.microsoft.com/resources/documentation/windows/2000/server/  
'scriptguide/en-us/sas_fil_lunl.msp  
'  
'Nécessite d'activer la référence Microsoft Shell Controls and Automation  
'  
Dim objShell As Shell32.Shell  
Dim strFileName As Shell32.FolderItem  
Dim objFolder As Shell32.Folder  
Dim Resultat As String, Reponse As String  
Dim i As Byte  
  
Set objShell = CreateObject("Shell.Application")  
'Répertoire cible  
Set objFolder = objShell.nameSpace("C:\Documents and Settings\michel")  
  
'boucle sur tous les elements du repertoire  
For Each strFileName In objFolder.Items  
  
    'Pour que les dosssiers ne soient pas pris en comptes  
    If strFileName.isFolder = False Then  
        Resultat = ""  
        For i = 0 To 34  
            If objFolder.getDetailsOf(strFileName, i) <> "" Then _  
                Resultat = Resultat & objFolder.getDetailsOf(objFolder.Items, i) _  
                    & ": " & objFolder.getDetailsOf(strFileName, i) & vbLf  
        Next  
        Reponse = MsgBox(Resultat & vbLf & vbLf & "Voulez vous continuer?", vbYesNo)  
        If Reponse = vbNo Then Exit Sub  
        End If  
    Next  
End Sub
```

Un autre exemple pour consulter un fichier spécifique.

```
Vb
```

```
Sub Test()  
    Dim sFich As String  
  
    sFich = "C:\Documents and Settings\michel\leClasseur.xls"  
    ListeProprietesFichier_getDetailsOf sFich  
  
End Sub  
  
Sub ListeProprietesFichier_getDetailsOf(Fichier As String)  
    'source:  
    'http://www.microsoft.com/resources/documentation/windows/2000/server/  
    'scriptguide/en-us/sas_fil_lunl.mspx  
    '  
    'Nécessite d'activer la référence Microsoft Shell Controls and Automation  
    '  
    Dim Fso As Object, oFichier As Object  
    Dim objShell As Shell32.Shell  
    Dim objFolder As Shell32.Folder  
    Dim strFileName As Shell32.FolderItem  
    Dim Chemin As String, NomFich As String, Resultat As String  
    Dim i As Byte  
  
    '-----  
    Set Fso = CreateObject("Scripting.FileSystemObject")  
    Set oFichier = Fso.GetFile(Fichier)  
    Chemin = Fso.GetParentFolderName(oFichier)  
    NomFich = Fso.GetFileName(oFichier)  
    '-----  
  
    Set objShell = CreateObject("Shell.Application")  
    Set objFolder = objShell.Namespace(Chemin)  
    Set strFileName = objFolder.Items.Item(NomFich)  
  
    For i = 0 To 34  
        'Cet exemple n'affiche pas les propriétés vides  
        If objFolder.GetDetailsOf(strFileName, i) <> "" Then _  
            Resultat = Resultat & objFolder.GetDetailsOf(objFolder.Items, i) _  
            & ": " & objFolder.GetDetailsOf(strFileName, i) & vbCrLf  
    Next  
  
    MsgBox Resultat  
End Sub
```

IV-D - Utiliser La classe CIM_DataFile

CIM_DataFile Représente un fichier dans le système d'exploitation Win32. Il s'agit d'une classe de Windows Management Instrumentation (WMI).

Pour plus de détails, consultez l'aide MSDN

Cette classe permet aussi de récupérer des informations sur les fichiers.

```
Vb
```

```
Sub Test()  
    Dim sFich As String
```

Vb

```
sFich = "C:\Documents and Settings\michel\leClasseur.xls"
Proprietes_CIM_Datafile sFich
End Sub

Sub Proprietes_CIM_Datafile(Fichier As String)
Dim strComputer As String
Dim objWMIService As Object, colFiles As Object, objFile As Object

strComputer = "."
Fichier = Replace(Fichier, "\", "\\")

Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set colFiles = objWMIService.ExecQuery _
    ("Select * from CIM_Datafile Where name = '" & Fichier & "'")

For Each objFile In colFiles
    Debug.Print "Archive: " & objFile.Archive
    Debug.Print "Description: " & objFile.Caption
    Debug.Print "Statut: " & objFile.Status
    'Valeurs possibles:
    'OK , Error , Degraded , Unknown , Pred Fail , Starting , Stopping , Service
    Debug.Print "Fichier système: " & objFile.System
    Debug.Print "Compression: " & objFile.Compressed
    Debug.Print "Methode de Compression: " & objFile.CompressionMethod
    Debug.Print "Date de creation: " & objFile.CreationDate
    Debug.Print "Lecteur: " & objFile.Drive
    Debug.Print "Chemin d'accès: " & objFile.EightDotThreeFileName
    Debug.Print "Cryptage: " & objFile.Encrypted
    Debug.Print "Méthode de cryptage: " & objFile.EncryptionMethod
    Debug.Print "Extension: " & objFile.Extension
    Debug.Print "Nom du fichier: " & objFile.FileName
    Debug.Print "Taille du fichier: " & objFile.FileSize & " octets"
    Debug.Print "Type du fichier: " & objFile.FileType
    Debug.Print "Système du fichier: " & objFile.FSName
    Debug.Print "Fichier caché: " & objFile.Hidden
    Debug.Print "Dernier accès: " & objFile.LastAccessed
    Debug.Print "Derniere modification: " & objFile.LastModified
    Debug.Print "Chemin complet du fichier: " & objFile.Name
    Debug.Print "Répertoire du fichier: " & objFile.path
    Debug.Print "Lisible: " & objFile.Readable
    Debug.Print "Version: " & objFile.Version
    Debug.Print "Modifiable: " & objFile.Writeable
Next
End Sub
```

V - Téléchargement