

# Lire et écrire dans les classeurs Excel fermés

par SilkyRoad ([silkyroad.developpez.com](http://silkyroad.developpez.com))

Date de publication : 30/10/2006

Dernière mise à jour : 26/08/2007

Ce tutoriel montre comment lire et écrire dans les classeurs Excel fermés en utilisant le modèle ADO.

Les exemples proposés ont été testés avec Excel2002, ADO 2.7 et ADOX 2.8

- I - Introduction
- II - Les limites d'Excel utilisé comme une base de données
- III - Les types de connexion
  - III-A - OLE DB Microsoft Jet
  - III-B - OLE DB pour pilotes ODBC
  - III-C - Se connecter aux classeurs Excel2007 xlsx et xlsm
- IV - Les requêtes
  - IV-A - Lire
  - IV-B - Ajouter un enregistrement
  - IV-C - Modifier les enregistrements
  - IV-D - Ajouter une feuille dans un classeur fermé
- V - Visualiser la structure des tables
- VI - Téléchargement

## I - Introduction

Ce tutoriel décrit les connexions et manipulations dans les classeurs Excel fermés.

Consultez les cours disponibles sur Developpez.com ( **Les fichiers et bases de données** ) pour obtenir plus de détails sur le modèle ADO.

Une partie des informations fournies dans ce document est issue de **l'aide en ligne Microsoft**.

Vous constaterez dans les exemples suivants que les requêtes sont identiques à celles utilisées dans les bases Access.

Les seules différences sont:

- \* Les limites spécifiques d'Excel (voir le chapitre II).
- \* Les syntaxes de connection (voir le chapitre III).
- \* La méthode d'identification des tables (les onglets ou plages de cellules nommées).

Bien entendu, Excel ne peut pas rivaliser avec Access en tant que base de données. Vous pouvez néanmoins travailler sur les classeurs de la même manière. Cette possibilité est particulièrement intéressante lorsque vous avez beaucoup de données et de nombreux classeurs à manipuler dans une même procédure. Le gain de temps peut alors être considérable.

Les informations fournies dans ce document supposent que le classeur est structuré comme une vraie base de données:

La première ligne sert à indiquer le nom des champs, à partir de la première colonne.

Les champs respectent les bonnes pratiques dans la déclaration des Noms:

- \* Nom le plus court possible.
- \* Pas d'espace.
- \* Pas d'accent.
- \* Pas de caractères spéciaux.

Évitez aussi les espaces et les caractères spéciaux dans le nom des feuilles.

Une table peut être une feuille de calcul ou une plage de cellules nommée.

Vous devez préalablement activer la référence [Microsoft ActiveX Data Objects x.x Library](#) pour utiliser les exemples présentés dans ce tutoriel.

Dans l'éditeur de macros:

Menu Outils.

Références.

Cochez la ligne "Microsoft ActiveX Data Objects x.x Library".

Cliquez sur le bouton OK pour valider.

x.x dépend de la version installée sur votre poste.

Certains exemples proposés permettent de manipuler les tables et nécessitent d'activer la référence [Microsoft ADO ext x.x for DLL and Security](#).

## II - Les limites d'Excel utilisé comme une base de données

Ce chapitre récapitule les limites et les opérations qui ne peuvent pas être réalisées dans les classeurs fermés.

Il n'est pas possible de supprimer les lignes complètes (enregistrements) dans un classeur fermé.

vous obtiendrez un message d'erreur "[La suppression des données dans un table attachée n'est pas géré par le pilote ISAM](#)".

Vous pouvez uniquement effacer le contenu des cellules.

Vous ne pouvez pas supprimer les lignes vides qui contenaient les données supprimées et les requêtes continueront d'afficher les enregistrements vides correspondant à ces lignes vides.

Il n'est pas possible de modifier une cellule contenant une formule.

Vous obtiendrez un message d'erreur "[L'opération demandée n'est pas autorisée dans ce contexte](#)".

Excel ne peut pas gérer les connections multiples et simultanées à un même classeur.

Les requêtes répétées peuvent entrainer des problèmes de mémoire dans Excel.

 **Consultez le site Microsoft pour plus de détails.**

Il n'est pas possible d'utiliser un classeur protégé par un mot de passe.

Il n'est pas possible d'utiliser le classeur si la feuille contenant les données est protégée.

Par défaut, le pilote ODBC analyse uniquement les 8 premières lignes du classeur fermé pour déterminer le type de données dans chaque colonne. Cela peut entrainer 2 types de problèmes:

1. Dans certains cas particuliers, les données exportées vers un classeur fermé peuvent être tronquées.

Si, par exemple, les 8 premiers enregistrements d'un champ contiennent des données texte inférieures ou égale à 255 caractères, le champ sera considéré de type Texte. Si ensuite vous ajoutez des enregistrements de longueur plus importante ils seront tronqués.



**Consultez le site Microsoft pour plus de détails.**

2. Si vous voulez importer les informations d'une colonne qui contient à la fois des données numériques et texte, c'est le type majoritaire dans les 8 premières lignes qui définira le type de données à récupérer: les autres données de la colonnes seront considérées comme NULL (vide).

Si la colonne contient 4 valeurs numériques et 4 valeurs texte, la requête renvoie 4 nombres et 4 valeurs NULL.

La seule solution consiste à activer l'option d'importation **IMEX=1**. Exemple: `"extended properties=""Excel 8.0;IMEX=1"""`.

Les données numériques seront importées comme du texte.

La taille des feuilles de calcul est limité à 65536 lignes par 256 colonnes...en attendant Excel2007...;o)

Le contenu des cellules (texte) est limité à 32767 caractères.

Le nombre de feuilles est limité par l'espace mémoire disponible.

### III - Les types de connexion

Il existe deux types de connexion disponibles pour lire et écrire dans un classeur fermé.

- \* OLE DB Microsoft Jet.
- \* OLE DB pour pilotes ODBC.

#### III-A - OLE DB Microsoft Jet

Ce premier exemple effectue une simple connexion au classeur puis la referme en utilisant le fournisseur OLE DB Microsoft Jet.

Il vous reste juste à adapter le chemin et le nom du fichier Excel.

Vba

```
Sub TestConnection_V1()  
    Dim Cn As ADODB.Connection  
    Dim Fichier As String  
  
    'Définit le classeur fermé servant de base de données  
    Fichier = "C:\monClasseurBase_V01.xls"  
  
    Set Cn = New ADODB.Connection  
  
    '--- Connexion ---  
    With Cn  
        .Provider = "Microsoft.Jet.OLEDB.4.0"  
        .ConnectionString = "Data Source=" & Fichier & _  
            ";Extended Properties=Excel 8.0;"  
        .Open  
    End With  
  
    'Extended Properties=Excel 8.0 est utilisé pour les versions d'Excel 97, 2000 et 2002.  
  
    '... la requête ...  
  
    '--- Fermeture connexion ---  
    Cn.Close  
    Set Cn = Nothing  
End Sub
```

#### III-B - OLE DB pour pilotes ODBC

La procédure suivante utilise le fournisseur OLE DB pour pilotes ODBC pour effectuer la connexion.

Vba

```
Sub TestConnection_V2()  
    Dim Cn As ADODB.Connection  
    Dim Fichier As String  
  
    'Définit le classeur fermé servant de base de données
```

Vba

```
Fichier = "C:\monClasseurBase_V01.xls"

Set Cn = New ADODB.Connection

With Cn
    .Provider = "MSDASQL"
    .ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
        "DBQ=" & Fichier & "; ReadOnly=False;"
    .Open
End With

'
'... la requête ...
'

'--- Fermeture connexion ---
Cn.Close
Set Cn = Nothing
End Sub
```

### III-C - Se connecter aux classeurs Excel2007 xlsx et xlsm

Utilisez la syntaxe suivante pour vous connecter aux classeurs Excel 2007, formats OpenXML xlsx et xlsm:

Vba

```
Sub RequeteClasseurFerme_Excel2007()
    Dim Cn As ADODB.Connection
    Dim Fichier As String
    Dim NomFeuille As String, texte_SQL As String
    Dim Rst As ADODB.Recordset

    'Définit le classeur fermé servant de base de données
    Fichier = "C:\Documents and Settings\mimi\dossier\NomClasseur.xlsx"
    'Nom de la feuille dans le classeur fermé
    NomFeuille = "Feuil1"

    Set Cn = New ADODB.Connection

    '--- Connexion ---
    With Cn
        .Provider = "Microsoft.Jet.OLEDB.4.0"
        .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & _
            & Fichier & ";Extended Properties=""Excel 12.0;HDR=YES;""
        .Open
    End With
    '-----

    '
    '... la requête ...
    '

    '--- Fermeture connexion ---
    Cn.Close
    Set Cn = Nothing

End Sub
```



## IV - Les requêtes

### IV-A - Lire

Avant de vous lancer dans l'exploration du modèle ADO, n'oubliez pas qu'Excel possède 2 outils afin de lire dans un classeur fermé:

Les formules de liaison:

```
= 'C:\Documents and Settings\mimi\dossier\excel\[ClasseurBase.xls]Feuill1'!$A$1
```

```
'Faire une RECHERCHE Verticale dans un classeur fermé:  
'Rechercher "DVP" dans la colonne A du classeur fermé et afficher la donnée correspondante de la  
colonne B.  
=RECHERCHEV("DVP";'C:\Documents and  
Settings\mimi\dossier\excel\[ClasseurBase.xls]Feuill1'!$A:$B;2;FAUX)
```

Les macros Excel4:

#### Vba

```
'Lecture de la cellule A1 dans la Feuill du classeur fermé  
MsgBox ExecuteExcel4Macro(" 'C:\Documents and  
Settings\mimi\dossier\excel\[ClasseurBase.xls]Feuill1'!R1C1")
```

Excel ne peut pas rivaliser avec Access mais les exemples suivants montrent qu'il est possible de manipuler les classeurs sur le même principe.

Voici une macro qui permet de se connecter à un classeur afin d'extraire le contenu de la feuille nommée "Feuil1".

#### Vba

```
Sub RequeteClasseurFermé()  
Dim Cn As ADODB.Connection  
Dim Fichier As String  
Dim NomFeuille As String, texte_SQL As String  
Dim Rst As ADODB.Recordset  
  
'Définit le classeur fermé servant de base de données  
Fichier = "C:\monClasseurBase.xls"  
'Nom de la feuille dans le classeur fermé  
NomFeuille = "Feuil1"  
  
Set Cn = New ADODB.Connection  
  
'--- Connection ---
```

Vba

```
With Cn
    .Provider = "Microsoft.Jet.OLEDB.4.0"
    .ConnectionString = "Data Source=" & Fichier & _
        ";Extended Properties=Excel 8.0;"
    .Open
End With
'-----

'Définit la requête.
'/*!\ Attention à ne pas oublier le symbole $ après le nom de la feuille.
texte_SQL = "SELECT * FROM [" & NomFeuille & "$]"

Set Rst = New ADODB.Recordset
Set Rst = Cn.Execute(texte_SQL)

'Ecrit le résultat de la requête dans la cellule A2
Range("A2").CopyFromRecordset Rst

'--- Fermeture connexion ---
Cn.Close
Set Cn = Nothing

End Sub
```

Ajoutez le paramètre HDR=NO si vous souhaitez également récupérer le contenu de la première ligne. Celle-ci est considérée comme un entête (ou Champ), par défaut.

```
";Extended Properties=""Excel 8.0;HDR=NO""
```

Etant donné que la première ligne est considérée comme un entête, il est aussi possible de boucler sur les noms de champs afin d'en extraire le contenu:

Vba

```
Dim i As Integer
'
'
'...
'
'Définit la requête.
texte_SQL = "SELECT * FROM [" & NomFeuille & "$]"

Set Rst = New ADODB.Recordset
Set Rst = Cn.Execute(texte_SQL)

'--- Boucle sur les entêtes pour récupérer les noms ---
For i = 0 To Rst.Fields.Count - 1
    Cells(1, i + 1) = Rst.Fields(i).Name
Next i
'-----

'Ecrit le résultat de la requête dans la cellule A2
Range("A2").CopyFromRecordset Rst
'
'...
'
```

Nota:

La méthode CopyFromRecordset permet d'afficher le résultat du Recordset dans la feuille de calcul, à partir de la cellule spécifiée (A2 dans l'exemple précédent).

### **La gestion des tables:**

Lors de la rédaction des requêtes, vous devez ajouter le symbole \$ à la suite du nom des onglets (les tables), ce qui n'est pas le cas des plages de cellules nommées (aussi considérées comme des tables).

Par contre si vous avez ajouté une table dynamiquement dans un classeur (en utilisant par exemple "Create Table" ou "SELECT INTO", comme dans le chapitre IV-D), deux noms différents sont renvoyés pour cette nouvelle table: avec et sans \$. En fait si vous ouvrez le classeur manuellement vous constaterez que l'onglet est bien ajouté mais aussi une plage de cellules nommée correspondant aux données insérées dynamiquement (voir le menu Insertion/Nom/Définir): Par exemple `=maNouvelleFeuille!$A$1:$C$1265`.

Le nom des feuilles doit être encadré par des crochets dans les requêtes `" & NomFeuille & "$]`.

Vous pouvez également lire une cellule spécifique, ou une plage de cellules, en utilisant la méthode ADO.

Vba

```
Sub extractionValeurCelluleClasseurFerme()  
    Dim Source As ADODB.Connection  
    Dim Rst As ADODB.Recordset  
    Dim ADOCommand As ADODB.Command  
    Dim Fichier As String, Cellule As String, Feuille As String  
  
    'Adresse de la cellule contenant la donnée à récupérer  
    Cellule = "B4:B4"  
    'Pour une plage de cellules, utilisez:  
    'Cellule = "A4:C10"  
  
    Feuille = "Feuil1$" 'n'oubliez pas d'ajouter $ au nom de la feuille.  
    'Chemin complet du classeur fermé  
    Fichier = "C:\Base.xls"  
  
    Set Source = New ADODB.Connection  
    Source.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
        "Data Source=" & Fichier & ";Extended Properties=""Excel 8.0;HDR=No;"";"  
  
    Set ADOCommand = New ADODB.Command  
    With ADOCommand  
        .ActiveConnection = Source  
        .CommandText = "SELECT * FROM [" & Feuille & Cellule & "]"  
    End With  
  
    Set Rst = New ADODB.Recordset  
    Rst.Open ADOCommand, , adOpenKeyset, adLockOptimistic  
  
    Set Rst = Source.Execute("[" & Feuille & Cellule & "]")
```

Vba

```
Range("A2").CopyFromRecordset Rst

Rst.Close
Source.Close
Set Source = Nothing
Set Rst = Nothing
Set ADODCommand = Nothing
End Sub
```

Ce dernier exemple montre comment effectuer une jointure entre 2 feuilles d'un classeur fermé.

La procédure compare 2 colonnes dans des onglets différents, et liste les données communes.

La Feuil1 contient une colonne dont l'entête s'appelle NumPeriode1.

La Feuil2 contient une colonne dont l'entête s'appelle NumPeriode2.

La macro liste les données de la Feuil2 qui apparaissent aussi dans la Feuil1.

Vba

```
Sub requeteJointure_ControleDoublons()
Dim Source As ADODB.Connection
Dim Requete As ADODB.Recordset
Dim Fichier As String, xSQL As String
Dim i As Long

Fichier = "C:\Base.xls"

Set Source = New ADODB.Connection
Source.Open "Provider = Microsoft.Jet.OLEDB.4.0;" & _
"data source=" & Fichier & ";" & _
"extended properties=" & "Excel 8.0;HDR=Yes"

xSQL = "SELECT DISTINCT [Feuil2$].NumPeriode2 " & "FROM [Feuil2$] " & _
"INNER JOIN [Feuil1$] ON [Feuil2$].NumPeriode2 = [Feuil1$].NumPeriode1"

Set Requete = New ADODB.Recordset
Set Requete = Source.Execute(xSQL)

If Requete.EOF Then
MsgBox "Il n'y a pas de doublons."
Else
'MsgBox "il y a des doublons."
Range("A2").CopyFromRecordset Requete
End If

Requete.Close
Source.Close
End Sub
```

## IV-B - Ajouter un enregistrement

Ce chapitre montre comment ajouter des enregistrements, à la suite des données existantes dans un classeur structuré comme une base de données. La première ligne sert à indiquer le nom des champs, à partir de la première colonne (A), et les enregistrements sont ensuite ajoutés automatiquement dans les lignes suivantes.

Les données seront insérées au format texte si aucun format de cellules n'est spécifié dans le fichier fermé.

Cet exemple ajoute un enregistrement dans la Feuil1 qui contient 4 champs (Date, Texte, Texte, Numérique).

Vba

```
Sub ajoutEnregistrement()  
    Dim Cn As ADODB.Connection  
    Dim Fichier As String, Feuille As String, strSQL As String  
    Dim LaDate As Date  
    Dim PrixUnit As Integer  
    Dim leNom As String, lePrenom As String  
  
    Fichier = "C:\Base.xls"  
    Feuille = "Feuill"  
  
    'Les données à insérer:  
    LaDate = CDate("26/05/2006")  
    leNom = "NomTest"  
    lePrenom = "PrenomTest"  
    PrixUnit = 40  
  
    Set Cn = New ADODB.Connection  
  
    With Cn  
        .Provider = "MSDASQL"  
        .ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _  
            "DBQ=" & Fichier & "; ReadOnly=False;"  
        .Open  
    End With  
  
    'Les données doivent être indiquées dans le même ordre que les champs dans la base de données.  
    strSQL = "INSERT INTO [" & Feuille & "$] " & _  
        & "VALUES (#" & LaDate & "#, " & _  
        "'" & leNom & "', " & _  
        "'" & lePrenom & "', " & _  
        PrixUnit & ")"  
  
    Cn.Execute strSQL  
  
    Cn.Close  
    Set Cn = Nothing  
End Sub
```

Le modèle ADO permet aussi d'écrire dans une cellule spécifique:

La macro suivante insère un texte dans la cellule G30 du classeur fermé.

Vba

```
Sub exportDonneeDansCelluleClasseurFerme()
```

## Vba

```
Dim Cn As ADODB.Connection
Dim Cd As ADODB.Command
Dim Rst As ADODB.Recordset
Dim Fichier As String

Fichier = "C:\Documents and Settings\mimi\dossier\LeClasseur.xls"

Set Cn = New ADODB.Connection
Cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=" & Fichier & ";" & _
    "Extended Properties=" & "Excel 8.0;HDR=No;" & ";"

Set Cd = New ADODB.Command
Cd.ActiveConnection = Cn
Cd.CommandText = "SELECT * FROM [Feuill$G30:G30]"

Set Rst = New ADODB.Recordset
Rst.Open Cd, , adOpenKeyset, adLockOptimistic
Rst(0).Value = "Donnée test"
Rst.Update

Cn.Close
Set Cn = Nothing
Set Cd = Nothing
Set Rst = Nothing
End Sub
```

## IV-C - Modifier les enregistrements

De la même manière que dans Access, vous pouvez effectuer des mises à jour dans la base en utilisant les requêtes UPDATE.

Cet exemple met à jour la valeur du "Champ4" si le "Champ2" correspond à la variable "leNom".

## Vba

```
Sub miseAJour_Enregistrement()
Dim Cn As ADODB.Connection
Dim Fichier As String, Feuille As String, strSQL As String
Dim PrixUnit As Integer
Dim leNom As String

Fichier = "C:\Base.xls"
Feuille = "Feuill"
leNom = "NomTest"
PrixUnit = 45

Set Cn = New ADODB.Connection

With Cn
.Provider = "MSDASQL"
.ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
    "DBQ=" & Fichier & ";" & "ReadOnly=False;"
.Open
End With

'Met à jour la valeur du "Champ4" si le "Champ2" correspond à la variable "leNom"
strSQL = "UPDATE [" & Feuille & "$] SET " & _
    "Champ4 = " & PrixUnit & " WHERE Champ2 = '" & leNom & "'"

```

Vba

```
Cn.Execute strSQL

Cn.Close
Set Cn = Nothing
End Sub
```

## IV-D - Ajouter une feuille dans un classeur fermé

Il est possible de créer dynamiquement une nouvelle feuille dans un classeur fermé en utilisant le modèle ADO.

Ce n'est pas quelque chose de très habituel mais il s'agit avant tout ici de montrer les possibilités associées à Excel.

Cet exemple montre comment ajouter une nouvelle feuille dans un fichier Excel fermé et y transférer le contenu d'une requête effectuée dans une table Access.

Vba

```
Sub tranfertTableAccess_Vers_ClasseurExcelFerme()
'Transfère une Table Access dans un nouvel onglet d'un classeur fermé.
'
Dim ExcelCn As ADODB.Connection
Dim ExcelRst As ADODB.Recordset
Dim AccessCn As New ADODB.Connection
Dim AccessRst As New ADODB.Recordset
Dim maBase As String, maFeuille As String, listeTable As String
Dim maTable As String, NomClasseur As String
Dim j As Integer
Dim Fld As ADODB.Field

'Chemin de la base Access
maBase = "C:\Documents and Settings\mimi\dossier\dataBase.mdb"
'Nom de la table Access à transférer
maTable = "Table1"
'Classeur où va être créée la nouvelle feuille
NomClasseur = "C:\leClasseurFermé.xls"
'Nom de la nouvelle feuille Excel
maFeuille = "MaNouvelleFeuille"

'Connection à la base Access
AccessCn.Open "provider=microsoft.jet.oledb.4.0; data source=" & maBase
'Requête dans la table Access
AccessRst.Open "SELECT * FROM " & maTable, AccessCn, adOpenStatic

'Connection au classeur Excel
Set ExcelCn = New ADODB.Connection
ExcelCn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=" & NomClasseur & ";" & _
            "Extended Properties=""Excel 8.0;HDR=NO;""

'paramétrage des entêtes et types de données
For Each Fld In AccessRst.Fields
    listeTable = listeTable & " " & FieldType(Fld.Type) & ","
Next Fld

'création nouvelle Feuille Excel
listeTable = Left(listeTable, Len(listeTable) - 1)
```

## Vba

```
ExcelCn.Execute "create table " & maFeuille & "(" & listeTable & ")"

Set ExcelRst = New ADODB.Recordset
ExcelRst.Open "Select * from " & maFeuille, ExcelCn, adOpenKeyset, adLockOptimistic

'transfert les données Access vers le classeur Excel
Do While Not (AccessRst.EOF)
    ExcelRst.AddNew
        For j = 0 To ExcelRst.Fields.Count - 1
            ExcelRst.Fields(j) = AccessRst.Fields(j).Value
        Next j
    ExcelRst.Update
    AccessRst.MoveNext
Loop

AccessRst.Close
AccessCn.Close
Set ExcelRst = Nothing
Set ExcelCn = Nothing
End Sub
```

## Vba

```
Function FieldType(Valeur As Long) As String
    'Spécification des types de données pour la création des champs Excel.
    'Attention ! la liste est incomplète.
    '
    Select Case Valeur
        Case 6
            FieldType = "currency"

        Case 7, 133, 134, 135
            FieldType = "Date"

        Case 14, 131
            FieldType = "Decimal"

        Case 5
            FieldType = "Float"

        Case 3, 2
            FieldType = "Integer"

        Case 4
            FieldType = "Real"

        Case 200, 202
            FieldType = "Text"

        Case 11
            FieldType = "Boolean"

        Case 203
            FieldType = "Memo"

        Case 16
            FieldType = "Tinyint"
    End Select
End Function
```

Une deuxième solution pour un résultat identique:

Vba

```
Sub tranfertTableAccess_Vers_ClasseurExcelFerme_V02()  
'Transfère une Table Access dans un nouvel onglet d'un classeur fermé  
,  
Dim ExcelCn As ADODB.Connection  
Dim ExcelRst As ADODB.Recordset  
Dim AccessCn As New ADODB.Connection  
Dim AccessRst As New ADODB.Recordset  
Dim maBase As String, maFeuille As String  
Dim maTable As String, NomClasseur As String  
Dim nbEnr As Long  
  
'Chemin de la base Access  
maBase = "C:\Documents and Settings\mimi\dossier\dataBase.mdb"  
'Nom de la table Access à transférer  
maTable = "Table1"  
'Classeur dans lequel va être créée la nouvelle feuille  
NomClasseur = "C:\leClasseurFermé.xls"  
'Nom de la nouvelle feuille Excel  
maFeuille = "MaNouvelleFeuille2"  
  
'Connection à la base Access  
AccessCn.Open "provider=microsoft.jet.oledb.4.0; data source=" & maBase  
'Requête dans la table Access  
AccessRst.Open "SELECT * FROM " & maTable, AccessCn, adOpenStatic  
  
'Connection au classeur Excel  
Set ExcelCn = New ADODB.Connection  
ExcelCn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=" & NomClasseur & ";" & _  
    "Extended Properties=" & "Excel 8.0;HDR=NO;" & ""  
  
'Transfert les données d'Access vers Excel  
AccessCn.Execute "SELECT * INTO [Excel 8.0;" & _  
    "Database=" & NomClasseur & "].[" & maFeuille & "] FROM " & maTable, nbEnr  
  
AccessRst.Close  
AccessCn.Close  
Set ExcelRst = Nothing  
Set ExcelCn = Nothing  
End Sub
```

## V - Visualiser la structure des tables

Si vous travaillez sur des fichiers Excel fermés, vous aurez peut être besoin de récupérer des informations sur leur structure: Lister le nom des feuilles, vérifier si une feuille spécifique existe, contrôler le format d'un champ...etc...Les modèles ADO et ADOX permettent d'obtenir ces données.

Tous les onglets et les plages de cellules nommées sont considérés comme des tables.

Vous pouvez lister leur nom en utilisant la macro suivante.

```
Vba
Sub Liste_Feuilles_PlagesNommees_ClasseurFerme_V01()
    'Nécessite d'activer la référence Microsoft ADO ext x.x for DLL and Security
    'Nécessite d'activer la référence Microsoft ActiveX Data Objects x.x Library
    Dim Cn As ADODB.Connection
    Dim oCat As ADOX.Catalog
    Dim Fichier As String, Resultat As String
    Dim Feuille As ADOX.Table

    Fichier = "C:\Base.xls"

    Set Cn = New ADODB.Connection
    Set oCat = New ADOX.Catalog

    Cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Fichier & _
        ";Extended Properties=Excel 8.0;"

    Set oCat.ActiveConnection = Cn

    For Each Feuille In oCat.Tables
        Resultat = Resultat & Feuille.Name & vbCrLf
    Next

    MsgBox Resultat

    Set Feuille = Nothing
    Set oCat = Nothing
    Cn.Close
    Set Cn = Nothing
End Sub
```

### Description du résultat de la macro:

- \* Les noms sont renvoyés par ordre alphabétique.
- \* Le nom des feuilles est suivi du symbole \$, ce qui n'est pas le cas des cellules nommées.
- \* Le nom des feuilles créées dynamiquement (voir le chapitre IV-D) n'ont pas le symbole \$.
- \* Le nom des feuilles créées dynamiquement est renvoyé 2 fois: Par l'onglet et la plage nommée associée.

\* Les noms qui contiennent un espaces sont encadrés par une quote '.

Voici une deuxième méthode pour lister le nom des tables.

Vba

```
Sub Liste_Feuilles_PlagesNommees_ClasseurFerme_V02()  
'Nécessite d'activer la référence Microsoft ActiveX Data Objects x.x Library  
Dim Cn As ADODB.Connection  
Dim rsT As ADODB.Recordset  
Dim Resultat As String, Fichier As String  
  
Fichier = "C:\Base.xls"  
  
Set Cn = New ADODB.Connection  
  
With Cn  
    .Provider = "Microsoft.Jet.OLEDB.4.0"  
    .ConnectionString = "Data Source=" & Fichier & _  
        ";Extended Properties=Excel 8.0;"  
    .Open  
End With  
  
Set rsT = Cn.OpenSchema(adSchemaTables)  
  
While Not rsT.EOF  
    Resultat = Resultat & rsT.Fields("TABLE_NAME") & vbCrLf  
    rsT.Fields("TABLE_NAME")  
    rsT.MoveNext  
Wend  
  
MsgBox Resultat  
  
rsT.Close  
Set rsT = Nothing  
Cn.Close  
Set Cn = Nothing  
End Sub
```

Vous pouvez aussi vérifier l'existence d'une feuille spécifique.

Vba

```
Sub VerifierExistenceFeuille()  
'Nécessite d'activer la référence Microsoft ADO ext x.x for DLL and Security  
'Nécessite d'activer la référence Microsoft ActiveX Data Objects x.x Library  
Dim Cn As ADODB.Connection  
Dim oCat As ADOX.Catalog  
Dim Fichier As String  
Dim Feuille As ADOX.Table  
  
Fichier = "C:\Base.xls"  
  
Set Cn = New ADODB.Connection  
Set oCat = New ADOX.Catalog  
  
Cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Fichier & _
```

Vba

```

        ";Extended Properties=Excel 8.0;"

    Set oCat.ActiveConnection = Cn

    On Error Resume Next
        'Vérifie si la feuille "Feuill" existe dans le classeur fermé
        Set Feuille = oCat.Tables("Feuill$")
    On Error GoTo 0

    If Feuille Is Nothing Then
        MsgBox "La feuille n'existe pas."
    Else
        MsgBox "La feuille existe."
    End If

    Set Feuille = Nothing
    Set oCat = Nothing
    Cn.Close
    Set Cn = Nothing
End Sub

```

Ce dernier exemple, issu de l'aide Microsoft, permet de visualiser la structure d'un classeur fermé.

Placez la procédure dans un UserForm et ajoutez y un contrôle ListBox nommé "ListBox1".

Vba

```

Private Sub UserForm_Initialize()
    '
    'Source: http://www.kbalertz.com/kb_Q257819.aspx#RetrieveMetadata
    '
    Dim Cn As ADODB.Connection
    Dim Rst As ADODB.Recordset, Rsc As ADODB.Recordset
    Dim intTblCnt As Integer, intTblFlds As Integer
    Dim strTbl As String, strCol As String
    Dim intColCnt As Integer, intColFlds As Integer
    Dim Fichier As String
    Dim t As Integer, c As Integer, f As Integer

    Fichier = "C:\Base.xls"

    Set Cn = New ADODB.Connection
    With Cn
        .Provider = "MSDASQL"
        .ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
            "DBQ=" & Fichier & ";"
        .Open
    End With

    Set Rst = Cn.OpenSchema(adSchemaTables)
    intTblCnt = Rst.RecordCount
    intTblFlds = Rst.Fields.Count

    ListBox1.AddItem "Tables: " & intTblCnt
    ListBox1.AddItem "-----"

    For t = 1 To intTblCnt

```

## Vba

```
strTbl = Rst.Fields("TABLE_NAME").Value
ListBox1.AddItem vbTab & "Table #" & t & ": " & strTbl
ListBox1.AddItem vbTab & "-----"

For f = 0 To intTblFlds - 1
    ListBox1.AddItem vbTab & Rst.Fields(f).Name & _
        vbTab & Rst.Fields(f).Value
Next

ListBox1.AddItem "-----"

Set Rsc = Cn.OpenSchema(adSchemaColumns, Array(Empty, Empty, strTbl, Empty))
intColCnt = Rsc.RecordCount
intColFlds = Rsc.Fields.Count

For c = 1 To intColCnt
    strCol = Rsc.Fields("COLUMN_NAME").Value

    ListBox1.AddItem vbTab & vbTab & "Column #" & c & ": " & strCol
    ListBox1.AddItem vbTab & vbTab & "-----"

    For f = 0 To intColFlds - 1
        ListBox1.AddItem vbTab & vbTab & Rsc.Fields(f).Name & _
            vbTab & Rsc.Fields(f).Value
    Next

    ListBox1.AddItem vbTab & vbTab & "-----"
    Rsc.MoveNext
Next

Rsc.Close
Set Rsc = Nothing

ListBox1.AddItem "-----"
Rst.MoveNext
Next

Rst.Close
Set Rst = Nothing
Cn.Close
Set Cn = Nothing
End Sub
```

## VI - Téléchargement

